UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UMA METODOLOGIA PARA O PROBLEMA DO CARTEIRO CHINÊS EM REDES MISTAS

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA
PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

JORGE RAUL BANEGAS CHAVEZ

FLORIANÓPOLIS - 1985

UMA METODOLOGIA PARA O PROBLEMA DO CARTEIRO CHINÊS EM REDES MISTAS

JORGE RAUL BANEGAS CHAVEZ

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE

"MESTRE EM ENGENHARIA"

ESPECIALIZADA EM ENGENHARIA DE PRODUÇÃO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO

Prof. Robert Wayne Samohyl Ph.D.
COORDENADOR DO PROGRAMA

BANCA EXAMINADORA

Prof. Raul Valentim da Silva M.Sc.
PRESIDENTE

Prof. Paulo R. Nascimento M.Sc.

Prof. Ricardo Barcia Ph.D.

AGRADECIMENTOS

Agradeço a todos aqueles que, de forma direta ou indireta, colaboraram para a realização deste trabalho, e em especial:

Ao Prof. Raul Valentim da Silva, que sempre se empenhou para que os problemas surgidos, durante a elaboração do trabalho, tivesse as melhores soluções;

Ao Prof. Paulo Renécio Nascimento, pelo seu magnífico trabalho de co-orientação, colaborando muito com suas idéias para o desenvolvimento do trabalho;

Ao meus Pais, pelo apoio e confiança sem limite, que sempre souberam dar;

A minha Esposa, pela paciência e dedicação, que sempre me brindou.

R E S U M O

O objetivo principal deste estudo, é o desenvolvimento de uma metodologia, que solucione o "Problema do Carteiro Chinês", em grafos mistos, visando aplicações diretas na distribuição de bens e serviços públicos. No conteúdo dessas aplicações, foram levantadas algumas restrições, associadas às leis de trânsito e manobras de veículos, quando essas se fazem necessárias.

São apresentadas soluções, que adaptam essas restrições, aos algoritmos e à rota final.

Com o intuito de alcançar, um melhor desempenho da metodologia proposta, foram pesquisados vários algoritmos, considerados, como os mais eficientes, nas questões envolvidas no problema.

Através da utilização destes algoritmos, e de procedimentos heurísticos, o grafo original, é modificado e transformado em um grafo euleriano, onde será aplicada a rota final.

Todos os algoritmos, relativos a esses procedimentos, e à rota final, são apresentados de uma forma estruturada, e para facilitar a sua compreensão e implementação, foram colocados alguns exemplos ilustrativos, nos casos em que foi julgado como necessário.

Nas conclusões, são apresentados, alguns aspectos, que podem ser abordados futuramente, visando uma posterior e se possível, melhor adaptação do trabalho ao problema real.

A B S T R A C T

The main purpose of this study, is to develop, a method of solving the "Chinese Postman Problem", in mixed graphs, aiming direct aplications in the distribution of assets and public services.

Due, to these aplications, some restrictions asso - ciated to the traffic laws and car maneuvering, were developed, when they were found necessary.

Solutions which adapt these restrictions to the algoritms related to the final route, are shown.

Trying to get a better performance, of the method, some algoritms related to the problem, were studied.

Using these algoritms and some heuristics procedures, the original graph is modified, and transformed in an eulerian graph, to find the final route.

All the algoritms related to these procedures and to the final route, are presented here in a structural way, and to simplify its comprehension and its implementation, some ilustrative examples were developed.

At the end, some aspects that could be approached in the future, are shown, aiming for a better adaptation of the work to the real problem.

SUMÁRIO

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

CAPÍTULO IV

CAPÍTULO V

CAPÍTULO VI

CAPÍTULO VII

CAPÍTULO I

1. INTRODUÇÃO

1.1. NOÇÕES FUNDAMENTAIS SOBRE GRAFOS

Nesta parte são apresentados alguns conceitos e definições importantes utilizadas no presente trabalho.

1.1.1. GRAFO OU REDE

É uma coleção de vértice, pontos ou nós X_1, X_2, ..., X_n (denotado pelo conjunto X) e uma coleção de linhas a_1, a_2, ..., a_n (denotado pelo conjunto A), unindo todos ou alguns destes pontos.[1]

O grafo está desta forma completamente descrito, denotando-o por G (X, A).

1.1.2. GRAFO ORIENTADO

Se todas as linhas têm direção, o que usualmente é mostrado por uma flecha, elas são chamadas de arcos e o grafo resultante é chamado de grafo orientado.[2]

---

[1] CHRISTOFIDES, Nicos - Graph Theory: An Algotithmic Approach. Academic Press, London, 1978. p. 1.

[2] Id.

1.1.3. GRAFO NÃO-ORIENTADO

Se todas as linhas estão sem orientação, então elas são chamadas de arestas e o grafo resultante é chamado de grafo não-orientado.[3]

Toda aresta pode ser substituída por dois arcos orientados, sendo representada por um segmento sem orientação ou por dois arcos de sentido opostos.

1.1.4. GRAFO MISTO

Denotado por G(X,A,E), é aquele grafo formado por três conjuntos de elementos:

X: conjunto finito de pontos, nós ou vértices.

$X = \{X_1, X_2, \ldots, X_n\}$

A: conjunto de arcos.

$A = \{(X_i, X_j)/(X_i, X_j) \in X \times X\}$

E: conjunto de arestas.

$E = \{(X_i, X_j), (X_j, X_i)/X_i, X_j \in X\}$

No caso em que um dos dois conjuntos seja vazio, o grafo torna-se um dos dois casos particulares definidos anteriormente. O grafo G(X, A) será chamado de grafo orientado e o grafo G(X, E) será denominado de grafo não-orientado.[4]

---

[3] CHRISTOFIDES, Nicos - Graph Theory: An Algorithmic Approach. Academic Press, London, 1978. p. 1.

[4] Id. p. 3.

Para o caso de grafo misto, as arestas também poderão ser representadas pelos seus dois arcos de sentidos opostos.

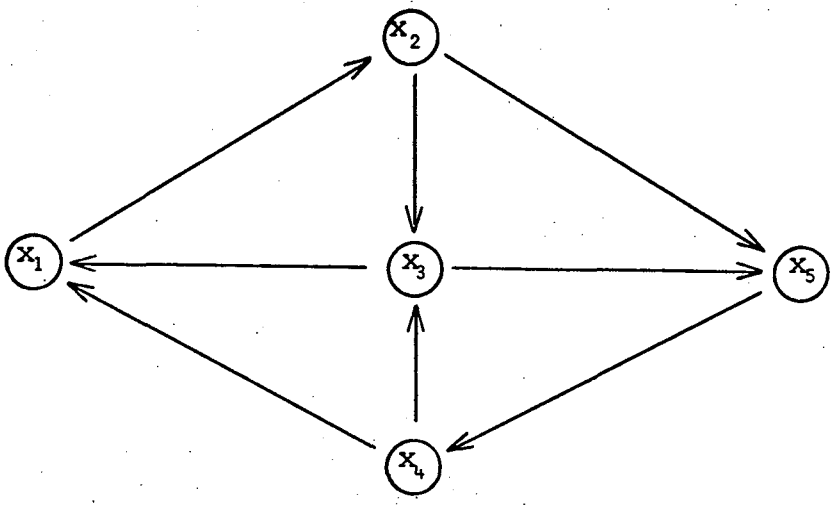


Fig. 1.1 Grafo orientado

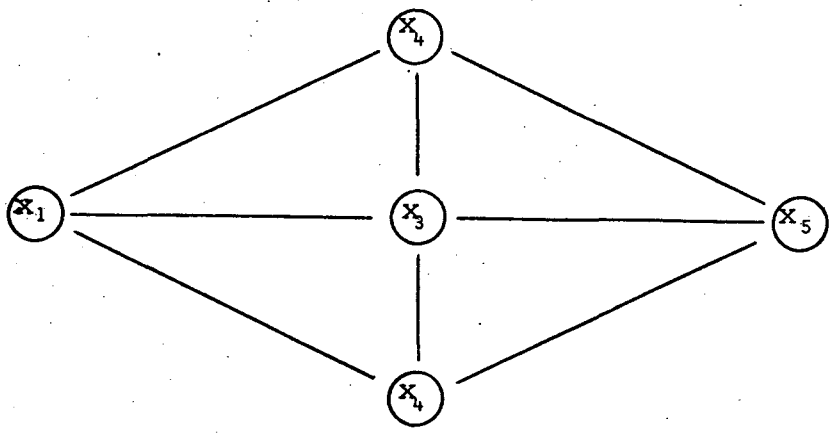


Fig. 1.2 Grafo não-orientado

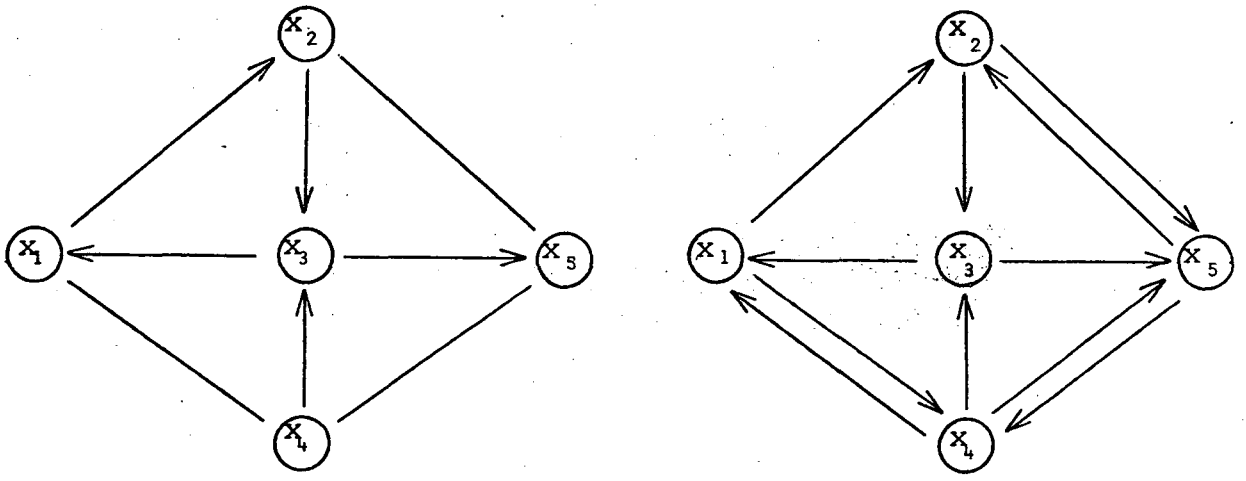


Fig. 1.3 Grafo misto

1.1.5. GRAFO VALORADO

Entende-se por grafo valorado, aquele ao qual pode-se atribuir valores aos vértices e/ou arcos.[5]

1.1.6. GRAFO BIPARTITE

Um grafo é bipartite se seus vértices puderem ser particionados em dois subconjuntos, X_1 e X_2, de tal modo que nenhuma linha seja incidente a dois vértices do mesmo subconjunto.[6]

1.1.7. SUB-GRAFO

Sub-grafo de G(X, A) é o grafo $G(N, A_n)$, onde $N \subset N$ e A_n é a familia de arcos de A que definidos pelo produto cartesiano $N \times N$.[7]

1.1.8. SUB-GRAFO PARCIAL

Sub-grafo parcial de G(X, A), é o grafo G(X', A'), onde $X' \subset X$ e $A' \subset A$. [8]

---

[5] FURTADO, A. Luz - Teoria dos Grafos: Algoritmos. Livros Técnicos e Científicos, Rio de Janeiro. p. 3.

[6] Id.

[7] Id. p. 5.

[8] Id. p. 7.

1.1.9. INCIDÊNCIA

Diz-se que um arco (ou aresta), é incidênte em um vértice, quando ele for um dos vértices desse arco (ou aresta).[9]

1.1.10. GRAU DE UM VÉRTICE

O grau de um vértice é o número de arcos nele incidentes. Utiliza-se a notação $d(x_i)$ para representar o grau do vértice x_i.[10]

1.1.11. GRAU DE ENTRADA

Num grafo orientado define-se como grau de entrada de um vértice x_i, e representa-se como $d_e(x_i)$, ao número total de arcos que têm o vértice x_i como seu vértice final.[11]

1.1.12. GRAU DE SAÍDA

Define-se como grau de saída de um vértice x_i, e representa-se como $d_s(x_i)$, ao número total de arcos que têm o

---

[9] FURTADO, A. Luz - Teoria dos Grafos: Algoritmos. Livros Técnicos e Científicos, Rio de Janeiro. p. 7.

[10] CHRISTOFIDES, Nicos - Graph Theory: An Algorithmic Approach, London, 1978. p. 7.

[11] Id.

vértice x_i como seu vértice inicial.[12]

1.1.13. DEMANDA

Denomina-se demanda de um vértice x_i, e representa-se por $D(x_i)$, à diferença entre o grau de saída e o grau de entrada desse vértice,[13] resultando:

$$D(x_i) = d_s(x_i) - d_e(x_i)$$

1.1.14. VÉRTICE PSEUDOSIMÉTRICO

Vértice pseudosimétrico é aquele que apresenta a propriedade de ter seu grau de entrada igual ao seu grau de saída.[14]

Num grafo orientado um vértice é dito de grau par, ou de grau ímpar, se seu número de arcos incidentes for par ou ímpar.

1.1.15. VÉRTICES FONTES

São aqueles cuja demanda é negativa denotando estes

---

[12] CHRISTOFIDES, Nicos - Graph Theory: An Algorithmic Approach, London, 1978. p. 7.

[13] MANDL, C - Applied Network Optimization, Academic Press London, 1979. p. 117.

[14] Id. (11). p. 346.

vértices fontes pela letra F, o conjunto F será dado por:

$$F = \{x_i / d_s(x_i) - d_e(x_i) < o\}^{15}$$

1.1.16. VÉRTICES SUMIDOUROS

São aqueles cuja demanda é positiva. Utilizando a letra S para representar esses vértices, o conjunto S será dado por:

$$S = \{x_j / d_s(x_j) - d_e(x_j) > o\}$$

1.1.17. VÉRTICES COMPLETOS

Denominam-se vértices completos aqueles cuja demanda é nula. Utilizando a letra C para representar este conjunto, resulta:

$$C = \{x_k / d_e(x_k) = d_s(x_k)\}$$

1.1.18. GRAFO PSEUDOSIMÉTRICO

É o grafo, no qual todos os seus vértices são pseudosimétricos.

---

[15] MANDL, C - Applied Network Optimization, Academic Press London, 1979. p. 117.

1.1.19. ASSOCIAÇÃO MÁXIMA

Um subconjunto M, do conjunto de arcos A, é uma associação se nenhum vértice do grafo for incidente em mais de um vértice. O subconjunto M de maior cardinalidade (isto é, com o maior número de elementos), que se puder formar em um grafo é uma associação máxima.[16]

1.1.20. SUPORTE DE UM GRAFO

Dado um subconjunto Z, do conjunto de vértices X de um grafo orientado, diz-se que Z, é um suporte, se cada arco do grafo possuir pelo menos uma extremidade em Z.[17]

1.1.21. CAMINHO

Um caminho em um grafo orientado, é qualquer sequência de arcos, onde o vértice final de um arco é o vértice inicial do próximo.

Um caminho simples é aquele que não utiliza o mesmo arco mais do que uma vez. Um caminho elementar é aquele que não utiliza o mesmo vértice mais do que uma vez.[18]

---

[16] FURTADO, A. Luz - Teoria dos Grafos: Algoritmos, Livros Técnicos e Científicos, Rio de Janeiro, 1973. p. 5.

[17] Id.

[18] CHRISTOFIDES, Nicos - Graph Theory: An Algorithmic Approach, Academic Press, London, 1978. p. 3-4.

1.1.22. CADEIA

Uma cadeia é uma sequência de arcos ou arestas de um grafo, tal que cada arco ou aresta tem:

1. uma extremidade em comum com o arco ou aresta antecedente (a exceção do primeiro).
2. a outra extremidade em comum com o arco ou aresta subsequênte (a exceção do último).

Como não se especifica de quais extremidades se trata, o conceito de cadeia é não orientado, por isso pode-se falar de uma cadeia constituída de arestas, em um grafo não orientado ou misto.[19]

Uma cadeia é dita simples se não usar duas vezes o mesmo arco ou aresta. É dita de elementar se não utilizar duas vezes o mesmo vértice.

1.1.23. CIRCUITO E CICLO

Um circuito é um caminho simples, no qual o vértice inicial e final coincidem, enquanto que um ciclo é uma cadeia simples na qual os vértices inicial e final se confundem.[20]

---

[19] CHRISTOFIDES, Nicos - Graph Theory: An Algorithmic Approach, Academic Press, London, 1978. p. 3-4.

[20] BOAVENTURA NETTO, P. O. - Teoria e Modelos de Grafos, E. Blucher, São Paulo, 1979. p. 24.

1.1.24. PERCURSO

É um termo genérico para caminhos, cadeias ciclos e circuitos.

Chama-se de percurso pré-euleriano, aquele percurso que utiliza toda aresta (ou todo arco), ao menos uma vez. Percurso euleriano é aquele que utiliza todo arco (ou toda aresta), uma vez e só uma.[21]

1.1.25. GRAFO EULERIANO

É aquele grafo que admite um percurso euleriano ou pré-euleriano.

1.1.26 GRAFO AUMENTADO

Um grafo aumentado é um grafo que foi alterado através da repetição de alguns arcos, e pelo direcionamento de algumas arestas, de forma a admitir um percurso euleriano ou pré-euleriano, transformando-se num grafo euleriano.

---

[21] BOAVENTURA NETTO, P. O. - Teoria e Modelos de Grafos, E. Blucher, São Paulo, 1979. p. 24.

1.2. OBJETIVO DO TRABALHO

Neste estudo procura-se, desenvolver uma metodolo - gia para encontrar a rota ótima numa rede mista, cujo custo total seja o menor possível.

A parte principal deste trabalho consiste em elaborar um algoritmo que consiga solucionar satisfatoriamente este problema.

O professor Edward Minieka, da Universidade de Chicago, apresentou uma metodologia para resolver o Problema do Carteiro Chinês para redes mistas quando existem vértices de grau ímpar.

Nesta metodologia, para cada aresta, adiciona-se um subgrafo à rede inicial de tal forma que a rede aumentada, pode ser resolvida pelo processo de fluxo de mínimo custo com ganhos.[22] A metodologia têm demonstrado ser eficiente, no sentido de encontrar uma solução para o problema do direcionamento das arestas da rede, mas observa-se que na medida que o número de arestas aumenta, o tempo de processamento torna-se proibitivo.

---

[22] MAURRAS, J - Optimization of the Flow Through Networks With Gains, Math. Programming, Volume 3 (1972). pp 135-144.

O mesmo problema é observado no método de Edmonds & Johnson.[23]

Isto é devido ao fato, de que ambas as metodologias pesquisam todas e cada uma das arestas da rede.

Objetiva-se, no presente trabalho, pesquisar aqueles vértices que apresentam uma demanda não nula e estabelecer uma atribuição entre esses vértices, de tal forma que na medida em que se for fazendo a atribuição, orientam-se simultaneamente as arestas.

Tomou-se como base para a análise deste problema particular, o fato de que toda aresta pode ser substituída por dois arcos de sentido contrário. Com esta consideração ao invés de ter uma rede mista, tem-se uma rede totalmente orientada, à qual poderão ser aplicados todos os teoremas sobre redes orientadas.

1.3. CONCEITOS BÁSICOS

Nesta seção serão explicados alguns procedimentos para a modificação do grafo original, que facilitarão a compreensão dos algoritmos apresentados nos capítulos subsequentes.

---

[23] EDMONDS, J. & JOHNSON, E. - Matching, Euler Tours and The Chinese Postman, Math, Programming, 5 (1973) pp 88-124.

1.3.1. MÉTODOS DE PROCESSAMENTO

Para o presente caso decidiu-se utilizar o método de processamento por matrizes. Embora ocupe bastante espaço de memória, apresenta a grande vantagem de ser extremamente versátil para trabalhar em linguagem Fortran.

A representação matricial mais usual de um grafo , sob a forma de matriz é dada pela matriz de adjacências.

Se $A = \{a_{i,j}\}$ é a matriz de adjacências de um grafo orientado G, então:

$a_{i,j} = 1$ se o arco (x_i, x_j) existe em G

$a_{i,j} = 0$ se o arco (x_i, x_j) não existe em G

Se o grafo não for orientado, a matriz será simetrica. Se o grafo for misto, então para cada aresta ter-se-a:

$$a_{i,j} = a_{j,i} = 1 \text{ se } (x_i, x_j) = (x_i, x_j)$$

A matriz de adjacências determina completamente um grafo.[24]

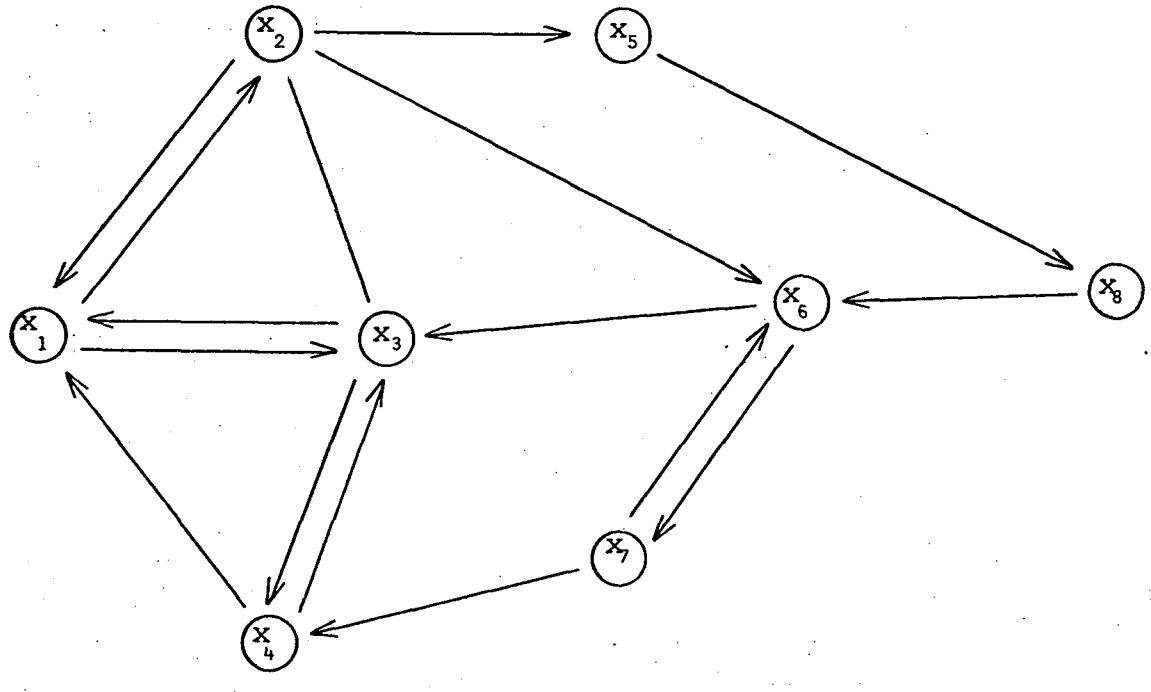
A matriz de adjacências apresenta algumas características muito úteis:

---

[24] CHRISTOFIDES, Nicos - Graph Theory: An Algorithmic Approach, Academic Press, London, 1978. p. 13.

1. a soma dos elementos na linha i da matriz fornece o grau de saída do vértice x_i;
2. a soma dos elementos na coluna j da matriz, fornece o grau de entrada do vértice x_j;
3. o conjunto de colunas que têm um valor 1 na linha i da matriz fornece os arcos subseqüentes (função de correspondência) ao vértice x_i;
4. o conjunto de linhas que têm um valor 1 na coluna j da matriz fornece a inversa da função de correspondência do vértice x_j.[25]

Seja o grafo misto representado pela fig. 1.4.



[25] CHRISTOFIDES, Nicos - Graph Theory: An Algorithmic Approach, Academic Press, London, 1978. p. 13.

A matriz de adjacência será:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	0	1	1	0	0	0	0	0
x_2	1	0	1	0	1	1	0	0
x_3	1	0	0	1	0	0	0	0
x_4	1	0	1	0	0	0	0	0
x_5	0	0	0	0	0	0	0	1
x_6	0	0	1	0	0	0	1	0
x_7	0	0	0	1	0	1	0	0
x_8	0	0	0	0	0	1	0	0

Neste caso, o grau de saída de x_1 é 2 e

o grau de entrada de x_1 é 3.

$\Gamma(x_1) = \{x_2, x_3\}$ significando que

de x_1 está saindo um arco para x_2 e um arco para x_3.

$\Gamma^{-1}(x_1) = \{x_2, x_3, x_4\}$ significa que em x_1 chega um arco que vem de x_2, outro que vem de x_3 e um tercei que vem de x_4.

As arestas (x_1, x_2), (x_1, x_3), (x_3, x_4) e (x_6, x_7) estão completamente descritas dado que:

$a_{1,2} = a_{2,1} = 1$

$a_{1,3} = a_{3,1} = 1$

$a_{3,4} = a_{4,3} = 1$, e

$a_{6,7} = a_{7,6} = 1$

A partir da matriz de adjacências pode-se calcular a demanda para cada vértice pela simples diferença entre o grau de entrada e o grau de saída do vértice. Pode-se determinar também os caminhos que convergem para cada vértice, pelo estabelecimento da função de correspondência e de sua inversa.

Para determinar se uma linha (x_i, x_j) pertence a G, basta verificar se $a_{i,j} = 1$. Se todos os elementos de A forem 1, G é um grafo completo.

Existem muitas outras características, que são facilmente verificáveis a partir da matriz de adjacências, mas não serão abordadas no presente trabalho por fugir ao escopo do mesmo.

1.3.2. CAMINHOS DE CUSTO MÍNIMO

Para um grafo valorado G(X, A), com custos dos arcos dados pela matriz $C = \{c_{i,j}\}$, o problema do caminho mínimo consiste em encontrar o caminho de menor custo, entre um vértice inicial $s \in X$ e um vértice final $t \in X$, caso este caminho exista.

É importante mencionar que os custos da matriz $\{c_{i,j}\}$ podem ser positivos, nulos ou negativos, desde que não exista em G nenhum circuito de custo negativo.

Existem diferentes casos que podem ser formulados como um problema de caminho de custo mínimo, em um grafo com n

vértices:

1. encontrar o caminho de custo mínimo entre dois vértices pré-definidos;
2. encontrar os caminhos de custo mínimo entre um vértice inicial s e todos os outros vértices do grafo;
3. encontrar os caminhos de custo mínimo entre todos os pares de vértices do grafo.

Todos estes casos podem ser resolvidos de uma maneira relativamente simples. Para o primeiro caso existem algoritmos eficientes que resolvem o problema. Com o mesmo algoritmo pode ser resolvido o segundo caso, fixando o vértice s inicial e alterando o vértice t final.

Finalmente, o terceiro caso pode ser resolvido pela aplicação n (nº de vértices) vezes de um algoritmo que resolva o caso número dois.

Um algoritmo (algoritmo de Floyd) para resolver este problema, será mais amplamente tratado nos capítulos seguintes.

1.3.3. CONCEITO DE ATRIBUIÇÃO

Uma vez conhecidos os vértices que são fontes, os vértices sumidouros e os caminhos de custo mínimo entre cada par de vértices do grafo, pode-se então estabelecer uma corres

pondência entre as fontes e os sumidouros, de tal forma que se fosse enviada uma unidade de fluxo desde uma fonte até um sumidouro, ter-se-ia a certeza de que o caminho percorrido por essa unidade de fluxo, seria o de menor custo possível.

Isto é o que se costuma chamar de atribuição. Como o critério utilizado é de minimização do custo associado a cada arco, pode-se dizer que desde o momento em que o algoritmo garanta uma solução ótima na atribuição, esta, poderá ser utilizada para fixar a orientação de uma aresta.

O problema da atribuição foi amplamente estudado em programação linear, podendo ser resolvido pelo algoritmo de transportes e até pelo método simplex, que encontra a solução ótima de todo e qualquer modelo de programação linear.

Seria entretanto, desperdício de tempo usar este método, uma vez que o algoritmo de atribuição explora a simplicidade daquele modelo, de uma forma mais eficiente.

1.3.4. ORIENTAÇÃO DAS ARESTAS

Até agora, não foi proposta nenhuma modificação na rede inicial. O estudo esteve limitado ao cálculo de demandas para cada vértice e, consequentemente, a uma classificação intrínseca desses vértices em vértices fontes, vértices completos e vértices sumidouros.

Foram caracterizados, também, os caminhos de custo mínimo entre cada par de vértices do grafo inicial e o grafo

bipartite. Com isto, conta-se agora com as ferramentas necessárias para a orientação ótima das arestas do grafo misto.

Baseados no fato de que qualquer vértice, seja fonte ou sumidouro, precisará da adição de um ou mais arcos, dependendo do valor de sua demanda, e de que para chegar de uma fonte até um sumidouro, percorre-se um e somente um caminho, resulta que os únicos arcos que precisarão ser modificados serão aqueles que estejam nesse percurso.

Utilizando sempre o critério de otimização do algoritmo de atribuição, as arestas que estejam nesse percurso, serão modificadas de acordo com a conveniência do algoritmo, chegando inclusive a anular um dos arcos da aresta (se necessário) ou a inverter o sentido de algum arco, obtendo-se depois desta modificação, dois arcos orientados no mesmo sentido.

Capítulo II

2. ALGORITMOS PARA A DETERMINAÇÃO DE CAMINHOS DE CUSTO MÍNIMO

Neste capítulo, será tratado o problema de selecionar dentre os caminhos existentes na rede, entre uma fonte e o sumidouro, aquele que apresenta o menor custo.

Para os objetivos deste trabalho, o importante é conhecer todos os caminhos de custo mínimo e as rotas associadas a estes. Levando em consideração este fato, foram pesquisados dois algoritmos, que são, até o presente momento, considerados os mais eficientes.

Estes algoritmos são: o algoritmo de Dijkstra e o algoritmo de Floyd. Será dado um tratamento suscinto ao algoritmo de Dijkstra e um tratamento mais detalhado ao algoritmo de Floyd, por ser o que melhor desempenho apresentou para os propósitos deste trabalho.

Para estes algoritmos, deve-se assumir que os custos dos arcos são maiores ou iguais a zero ($c_j \geq 0$), para todos os arcos que pertençam ao conjunto A. No presente trabalho, isto não representa uma consideração restritiva, dado que os custos negativos não representam um significado prático especial nos problemas considerados.

2.1. ALGORITMO DE DIJKSTRA[25]

Foi desenvolvido originalmente para grafos finitos com custos positivos, situação em que é admissível.

Se o grafo G(X, A) tem N vértices e nenhum arco tem custo negativo, este algoritmo fecha no máximo N nós antes de achar a solução ótima, ou concluir pela inexistência de uma. Se se admitir a existência de custos negativos, mas não de circuitos com custos negativos, uma modificação no algoritmo transforma-o novamente em admissível.

2.1.1. INICIALIZAÇÃO

1. Para cada nó $x \in X$ guarda-se a sua descrição;
2. define-se dois conjuntos de nós abertos A e Fechados F;
3. Cria-se um conjunto de apontadores P(x), inicialmente vazio;
4. Cria-se um conjunto de custos $\hat{g}(x)$, que representa o custo do melhor caminho já encontrado entre o vértice inicial s e o vértice x.

2.1.2. ALGORITMO PROPRIAMENTE DITO

1. A = s; $\forall s \in S$ faça $\hat{g}(s) = 0$, P(s) = 0;
   F = $\emptyset$

2. Se $A = \emptyset$, pare com fracasso. Do contrário, tome x tal que $\hat{g}(x) = \min \hat{g}(u)$; se houver mais de um, desempate de qualquer modo, mas sempre a favor de nós em T.

3. Faça $A = A - \{x\}$, $F = F \cup \{x\}$. Se $x \in T$, pare com sucesso. Do contrário gere T(x); se $T(x) = \emptyset$ volte a 2.

4. Para cada $m \in T(x)$, faça $\hat{f} = \hat{g}(x) + C_{xm}$.

   Se ($m \in A$ e $\hat{g}(m) \leq \hat{f}$) ou ($m \in F$), tome o próximo sucessor e refaça o teste. Do contrário, faça $\hat{g}(m) = \hat{f}$, $P(m) = x$ e $A = A \cup \{m\}$.

5. Volte ao passo 2.

2.2 ALGORITMO DE FLOYD PARA ENCONTRAR OS CAMINHOS DE CUSTO MÍNIMO E AS ROTAS ASSOCIADAS[26]

Este algoritmo está baseado na modificação iterativa de matrizes formadas a partir da matriz de custos associada a uma rede.

Cada matriz gerada possui custos menores ou no máximo iguais aos seus correspondentes anteriores. Com isto o al-

---

[26] BOAVENTURA NETTO, Paulo O., Teoria e Modelos de Grafos. E. Blucher, São Paulo, 1979. pp 167-171.

goritmo pesquisa novos caminhos, comparando-se com os já analisados. No final o algoritmo fornece uma matriz, que dá os caminhos de custo mínimo entre dois vértices quaisquer do grafo, e outra matriz que permite encontrar as rotas associadas a estes caminhos.

Parte-se de uma matriz $C = \{c_{i,j}\}$ de custos dos arcos, na qual se indicam os custos infinitos para os arcos inexistentes.

O algoritmo constroi, sucessivamente, n matrizes a partir da matriz C, através de modificações efetuadas em acordo com a seguinte expressão:

$$c^{k}_{i,j} = \min \{c^{k-1}_{i,j},\ (c^{k-1}_{i,k} + c^{k-1}_{k,i})\}$$

onde se varrem, nesta ordem, i, j e k.

Na iteração k, trabalha-se com a matriz C^{k-1} e se varrem a linha k e a coluna k dessa matriz. Pode não ocorrer modificação em uma iteração qualquer.

A matriz que permite encontrar as rotas associadas é, geralmente chamada "matriz de Roteamento", é uma forma conveniente de apresentação de todos os caminhos, obtidos pela aplicação de um algoritmo matricial. Também é denominada matriz de uniroteamento, uma vez que só permite a descrição de um caminho para cada par de vértices.

Sua construção é baseada no teorema correspondente ao

"Princípio de Bellman"[27], válido para grafos valorados.

Sendo esta matriz $D = \{d_{i,j}\}$, tem-se

$$d_{i,j} = i$$

$$d_{i,j} = k \quad x_k \varepsilon\ T(x_i) \text{ e } x_k \varepsilon\ M_{i,j}.$$

Portanto, k é o índice do vértice imediatamente seguinte na sucessão dada pelo caminho. $M_{i,j} = (x_i, x_k, \ldots, x_t, x_j)$ produzirá:

$d_{i,j} = k$, $d_{k,j} = \ldots, \ldots$, $d_{t,j} = j$ que se encerra ao ser obtido o índice do vértice final do caminho. (Se $d_{i,j,} = j$, $i \neq j$, o caminho possui um arco único).

Para a descrição do algoritmo utiliza-se a seguinte notação:

$c_{i,j}$ = custo associado ao arco que vai do vértice i até o vértice j, $i,j \varepsilon X$. Este custo terá um valor infinito se o arco (i,j) não existir.

---

[27] BOAVENTURA NETTO, Paulo O., Teoria e Modelos de Grafos. E. Blucher, São Paulo, 1979. p. 77.

2.2.1. ALGORITMO[28]

Passo 1 inicialização

fazer $k = 0$

fazer $d_{i,j} = i$ para todo $x_i, x_j \varepsilon X$

Passo 2 iteração

fazer $k = k + 1$

fazer $c_{i,j} = \min \{c_{i,j}, (c_{i,k} + c_{k,j})\}$

para todo $i \neq k$ de tal forma que $c_{i,k} \neq \infty$ e todo $j \neq k$ de tal forma que $c_{k,j} \neq \infty$ se $(c_{i,k} + c_{k,j}) < c_{i,j}$ então fazer $d_{i,j} = d_{k,j}$

Passo 3 final

Se qualquer $c_{j,j} < o$ então existe um circuito negativo contendo o vértice j e não existe solução possível, o algoritmo termina.

Se $k = n$ e se $c_{j,j} > 0$, o algoritmo termina com a solução desejada.

Se $k < n$ volta-se ao passo 2

---

[28] MANDL, Christoph, Applied Network Optimization, Academic Press, London, 1979. pp 9-14.

No final do algoritmo os valores $c_{i,j}$ corresponderão ao caminho de custo mínimo entre o vértice i e o vértice j, e o valor $c_{j,j}$ corresponderá ao circuito de custo mínimo que passa pelo vértice j. As rotas associadas podem ser encontradas facilmente a partir da matriz D.

Para ilustrar melhor o algoritmo, será desenvolvido um exemplo, supondo que se tenha a rede valorada da fig. 2.1

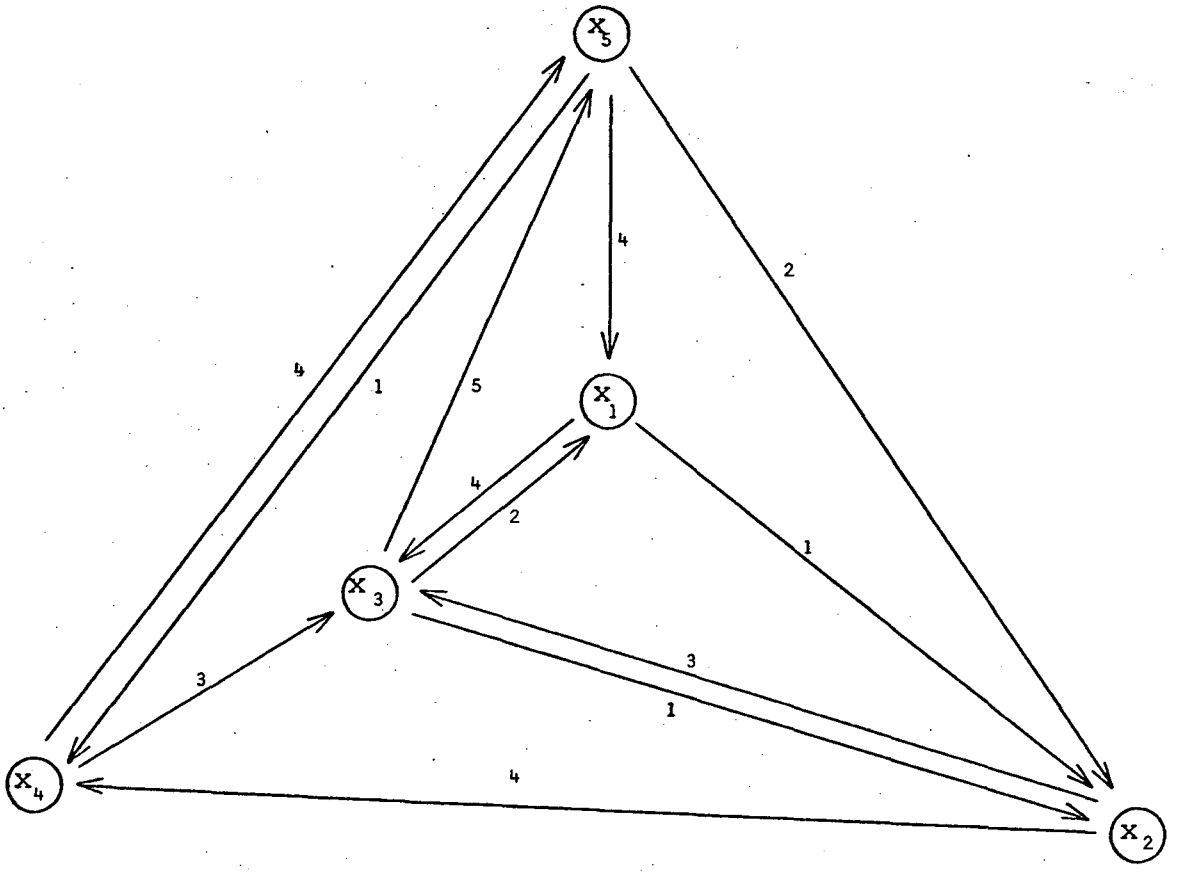


Fig. 2.1. Um exemplo ilustrativo

com a matriz de adjacências dada por:

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

e com matriz de custos associados, dada por:

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} \infty & 1 & 4 & \infty & \infty \\ \infty & \infty & 3 & 4 & \infty \\ 2 & 1 & \infty & \infty & 5 \\ \infty & \infty & 3 & \infty & 4 \\ 4 & 2 & \infty & 1 & \infty \end{bmatrix} \end{array}$$

A aplicação do algoritmo resultará em:

Passo 1 $\quad k = 0$

$$C = \begin{bmatrix} \infty & 1 & 4 & \infty & \infty \\ \infty & \infty & 3 & 4 & \infty \\ 2 & 1 & \infty & \infty & 5 \\ \infty & \infty & 3 & \infty & 4 \\ 4 & 2 & \infty & 1 & \infty \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

Passo 2 $\quad k = 1$

$$C = \begin{bmatrix} \infty & 1 & 4 & \infty & \infty \\ \infty & \infty & 3 & 4 & \infty \\ 2 & 1 & 6 & \infty & 5 \\ \infty & \infty & 3 & \infty & 4 \\ 4 & 2 & 8 & 1 & \infty \end{bmatrix} \qquad D = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 1 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 1 & 5 & 5 \end{bmatrix}$$

Amostra de cálculo:

$$c_{3,3,} = \text{mín}\ \{c_{3,3},\ (c_{3,1} + c_{1,3})\}$$

$$= \text{mín}\ \{\ \infty\ ,\ (2\quad +\quad 4)\} = 6;$$

$$d_{3,3} = d_{1,3}$$

$$= 1$$

$$c_{5,3} = \text{mín}\ \{c_{5,3},\ (c_{5,1} + c_{1,3})\}$$

$$= \text{mín}\ \{\ \infty\ ,\ (4\quad +\quad 4\ )\} = 8;$$

$$d_{5,3} = d_{1,3}$$
$$= 1$$

Passo 3 Volta-se ao passo 2

Passo 2 k = 2

$$C = \begin{pmatrix} \infty & 1 & 4 & 5 & \infty \\ \infty & \infty & 3 & 4 & \infty \\ 2 & 1 & 4 & 5 & 5 \\ \infty & \infty & 3 & \infty & 4 \\ 4 & 2 & 5 & 1 & \infty \end{pmatrix} \qquad D = \begin{pmatrix} 1 & 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 2 & 2 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 2 & 5 & 5 \end{pmatrix}$$

Passo 3 Volta-se ao passo 2

Passo 3 K = 3

$$C = \begin{pmatrix} 6 & 1 & 4 & 5 & 9 \\ 5 & 4 & 3 & 4 & 8 \\ 2 & 1 & 4 & 5 & 5 \\ 5 & 4 & 3 & 8 & 4 \\ 4 & 2 & 5 & 1 & 10 \end{pmatrix} \qquad D = \begin{pmatrix} 3 & 1 & 1 & 2 & 3 \\ 3 & 3 & 2 & 2 & 3 \\ 3 & 3 & 2 & 2 & 3 \\ 3 & 3 & 4 & 2 & 4 \\ 5 & 5 & 2 & 5 & 3 \end{pmatrix}$$

Passo 3 Volta-se ao passo 2

Passo 2 K = 4

$$C = \begin{bmatrix} 6 & 1 & 4 & 5 & 9 \\ 5 & 4 & 3 & 4 & 8 \\ 2 & 1 & 4 & 5 & 5 \\ 5 & 4 & 3 & 8 & 4 \\ 4 & 2 & 4 & 1 & 5 \end{bmatrix} \qquad D = \begin{bmatrix} 3 & 1 & 1 & 2 & 3 \\ 3 & 3 & 2 & 2 & 3 \\ 3 & 3 & 2 & 2 & 3 \\ 3 & 3 & 4 & 2 & 4 \\ 5 & 5 & 4 & 5 & 4 \end{bmatrix}$$

Passo 3 Volta-se ao passo 2

Passo 2 K = 5

$$C = \begin{bmatrix} 6 & 1 & 4 & 5 & 9 \\ 5 & 4 & 3 & 4 & 8 \\ 2 & 1 & 4 & 5 & 5 \\ 5 & 4 & 3 & 5 & 5 \\ 4 & 2 & 4 & 1 & 5 \end{bmatrix} \qquad D = \begin{bmatrix} 3 & 1 & 1 & 2 & 3 \\ 3 & 3 & 2 & 2 & 3 \\ 3 & 3 & 2 & 2 & 3 \\ 3 & 3 & 4 & 5 & 4 \\ 5 & 5 & 4 & 5 & 4 \end{bmatrix}$$

Passo 3 Termina o algoritmo

Desta forma, o caminho de custo mínimo, por exemplo, entre o vértice 4 e o vértice 1 custará 5 unidades monetárias, e a rota associada a ele será a seguinte:

$$x_4 - x_3 - x_1.$$

Capítulo III

3. ALGORITMO DE ATRIBUIÇÃO

Neste capítulo será tratado o problema da atribui - ção no grafo bipartite, criado com a classificação dos vérti ces, em fonte e sumidouros.

Para resolver este problema, foi selecionado o Algo ritmo Húngaro, pela sua simplicidade no manuseio dos dados.

3.1. ALGORITMO HÚNGARO

O algoritmo Húngaro trabalha sobre a matriz $C = \{c_{i,j}\}$ de custos associados ao grafo bipartite formado pelo vértices fonte e pelos vértices sumidouros. Um exemplo deste grafo apa rece na fig. 3.1, assim como sua respectiva matriz de custos, que poderiam ser obtidos pela aplicação do algoritmo de Floyd. O funcionamento deste algoritmo baseia-se no seguinte lema:

> "não se altera a solução ou soluções ótimas do pro- blema de atribuição, diminuindo ou aumentando de uma mesma quantidade λ todos os elementos de uma linha ou coluna. Tal operação apenas diminui ou aumenta de λ o valor total, mas não afeta a esco - lha dos arcos"[29]

---

[29] FURTADO, Antonio L. Teoria dos Grafos: Algoritmos. Livros Técnicos e Científicos, Rio de Janeiro, 1973, pp. 73.

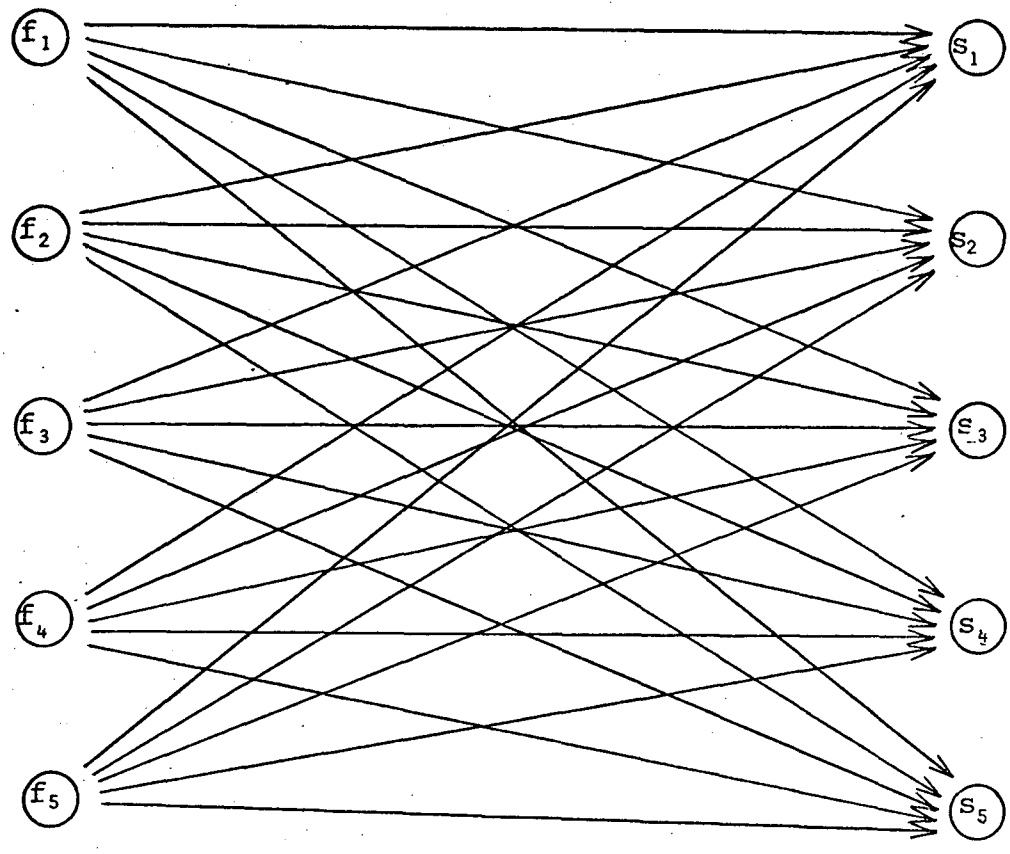


Fig. 3.1. Grafo bipartite

	s_1	s_2	s_3	s_4	s_5
f_1	7	3	5	7	10
f_2	6	∞	∞	8	7
f_3	6	5	1	5	∞
f_4	11	4	∞	11	15
f_5	∞	4	5	2	10

Matriz de custos associados ao grafo da fig. 3.1

3.1.1. FASE I. OBTENÇÃO DE ZEROS

A todos os elementos de cada coluna substrair o menor elemento da coluna; da matriz resultante substrair a todos os elementos de cada linha, o menor elemento de uma linha.

Assim é garantida a obtenção de pelo menos um zero em cada linha e em cada coluna.

Para o grafo da fig. 3.1, tomado como exemplo, a matriz de custos fica:

	s_1	s_2	s_3	s_4	s_5
f_1	1	0	4	5	3
f_2	0	∞	∞	6	0
f_3	0	2	0	3	∞
f_4	4	0	∞	8	7
f_5	∞	1	4	0	3

3.1.2. FASE II. PESQUISA DA SOLUÇÃO ÓTIMA

Com os zeros obtidos na fase anterior, tenta-se formar uma solução de valor total zero. Se isso for possível termina o algoritmo, do contrário passa-se a fase seguinte.

Para a pesquisa dessa solução nula, considera-se de início uma das linhas que tenha o menor número de zeros, assi

nala-se um dos zeros dessa linha e cancelam-se os zeros que se encontrem na mesma linha ou coluna que o zero assinalado . Repete-se o procedimento para todas as linhas.

No exmplo vê-se que não foi possível obter uma solução nula:

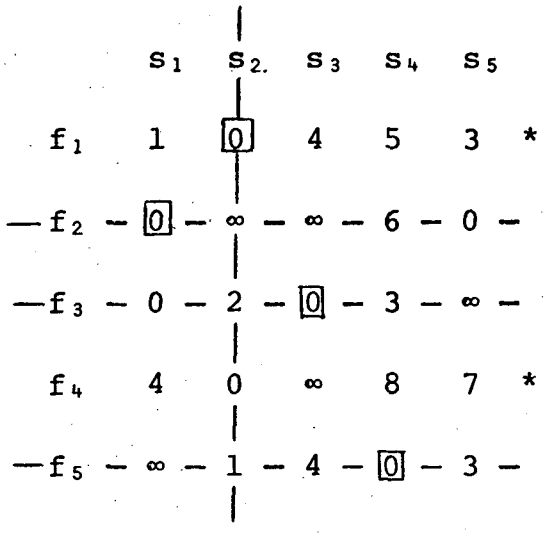
	s_1	s_2	s_3	s_4	s_5
f_1	1	[0]	4	5	3
f_2	[0]	∞	∞	6	0
f_3	0	2	[0]	3	∞
f_4	4	0	∞	8	7
f_5	∞	1	4	[0]	3

3.1.3. FASE III. OBTENÇÃO DE UM CONJUNTO MÍNIMO DE LINHAS E COLUNAS CONTENDO TODOS OS ZEROS

1. Marcar com asterisco todas as linhas que não contenham nenhum zero assinalado.
2. Marcar toda coluna que contenha um ou mais zeros cancelados em uma das linhas marcadas.
3. Marcar toda linha que contenha um zero assinalado em uma coluna marcada.
4. Repetir os passos 2 e 3 até que não seja mais possível marcar outras linhas ou colunas.

5. Colocar um traço sobre toda linha não marcada e também sobre toda coluna marcada.

Desta forma o exemplo fica:



3.1.4. FASE IV. DESLOCAMENTO EVENTUAL DE CERTOS ZEROS

Tomar o menor número não cortado por um traço, subtraí-lo dos elementos das linhas não cortadas, e somá-lo aos elementos das colunas cortadas.

Para o exemplo, é tomado o elemento $c_{1,1}$:

	s_1	s_2	s_3	s_4	s_5
f_1	0	0	3	4	2
f_2	0	∞	∞	6	0
f_3	0	3	0	3	∞
f_4	3	0	∞	7	6
f_5	∞	2	4	0	3

Em seguida volta-se à fase II até atingir a solução ótima (que pode não ser única).

Para o exemplo a solução final será:

	s_1	s_2	s_3	s_4	s_5
f_1	[0]	0	3	4	2
f_2	0	∞	∞	6	[0]
f_3	0	3	[0]	3	∞
f_4	3	[0]	∞	7	6
f_5	∞	2	4	[0]	3

Com relação à matriz inicial, pode-se verificar que essa solução corresponde ao custo:

$C_{1,1} + C_{2,5} + C_{3,3} + C_{4,2} + C_{5,4} = 21$ e a solução do problema está dada na figura 3.2

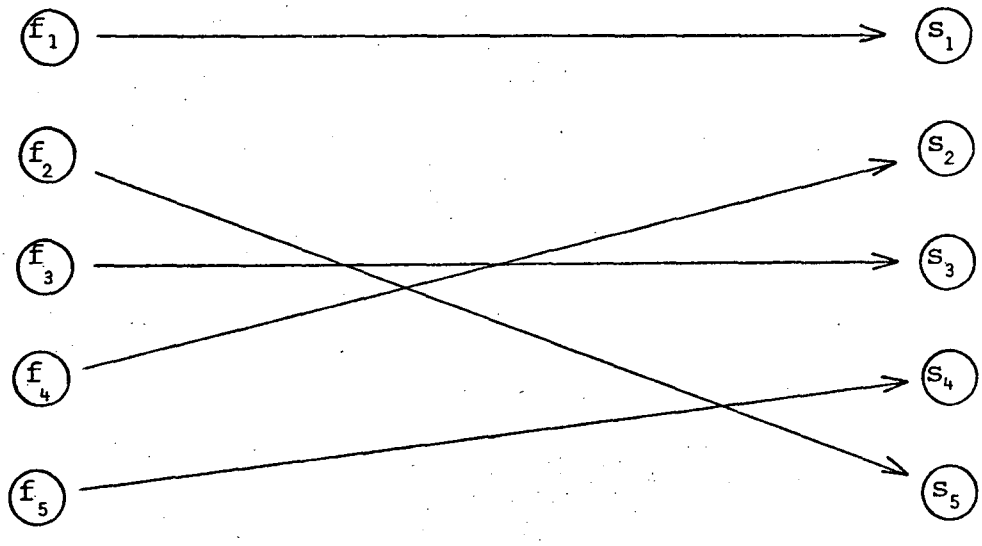


Fig. 3.2 Solução do grafo bipartite da fig. 3.1

3.2. ORIENTAÇÃO DOS ARESTAS

Até agora não se tinha ferramentas para poder orientar as arestas da rede. Agora já sabendo quais são as fontes, quais são os sumidouros, bem como os caminhos de custo mínimo, e já feita a atribuição entre cada fonte e cada sumidouro do grafo, pode-se mandar uma unidade de fluxo desde uma fonte até um sumidouro. Esta unidade, estará na realidade orientando as arestas.

Desta maneira, estar-se-á considerando somente as arestas correspondentes, reduzindo com isto o trabalho de pesquisa.

Para ilustrar, supondo que se tenha o grafo da fig. 3.3

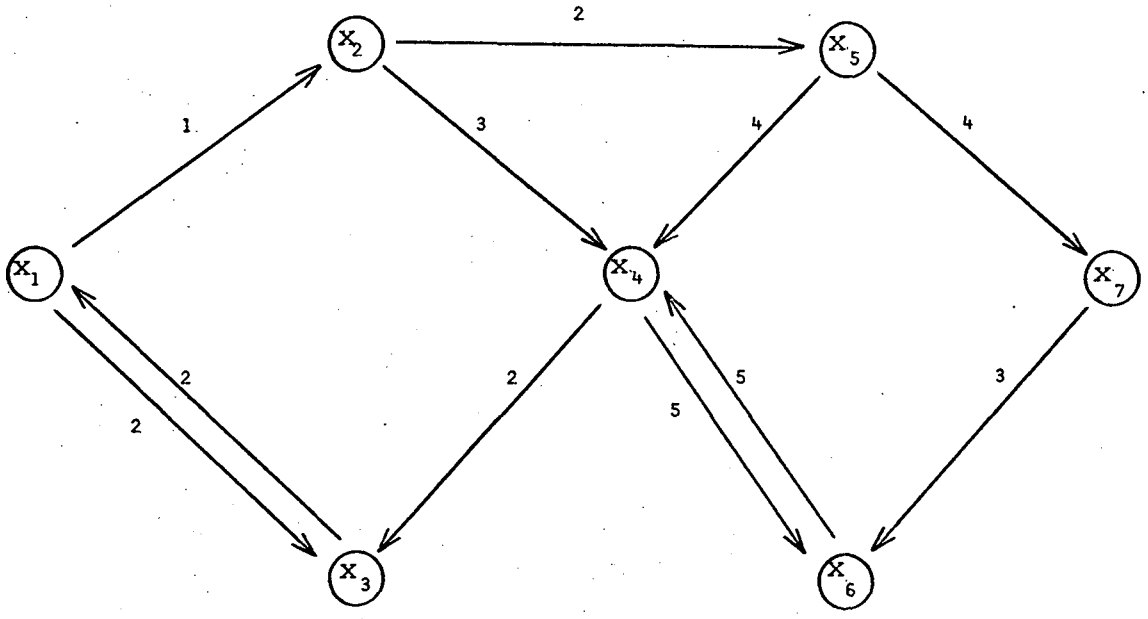


Fig. 3.3 Exemplo Ilustrativo

O cálculo das demandas normais seria o seguinte:

Vértice	Número de Saídas	Número de Entradas	Demanda
x_1	2	1	1
x_2	2	1	1
x_3	1	2	-1
x_4	2	3	-1
x_5	2	1	1
x_6	1	2	-1
x_7	1	1	0

Tabela 3.1. Cálculos do exemplo 3.3

A matriz de custos seria:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_1	0	1	2	∞	∞	∞	∞
x_2	∞	0	∞	3	2	∞	∞
x_3	2	∞	0	∞	∞	∞	∞
x_4	∞	∞	2	0	∞	5	∞
x_5	∞	∞	∞	4	0	∞	4
x_6	∞	∞	∞	5	∞	0	∞
x_7	∞	∞	∞	∞	∞	3	0

E a matriz de custo mínimo seria, aplicando o algoritmo de Floyd:

$$
\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\
\left(\begin{array}{ccccccc}
0 & 1 & 2 & 4 & 3 & 9 & 7 \\
7 & 0 & 5 & 3 & 2 & 8 & 6 \\
2 & 3 & 0 & 6 & 5 & 11 & 9 \\
4 & 5 & 2 & 0 & 7 & 5 & 11 \\
8 & 9 & 6 & 4 & 0 & 7 & 4 \\
9 & 10 & 7 & 5 & 12 & 0 & 16 \\
12 & 13 & 10 & 8 & 15 & 3 & 0
\end{array}\right)
\end{array}
$$

De acordo com as demandas calculadas, o conjunto de fontes será $\{x_3, x_4 \text{ e } x_6\}$ e o conjunto de sumidouros será $\{x_1, x_2 \text{ e } x_5\}$ resultando o grafo bipartite da fig. 3.4

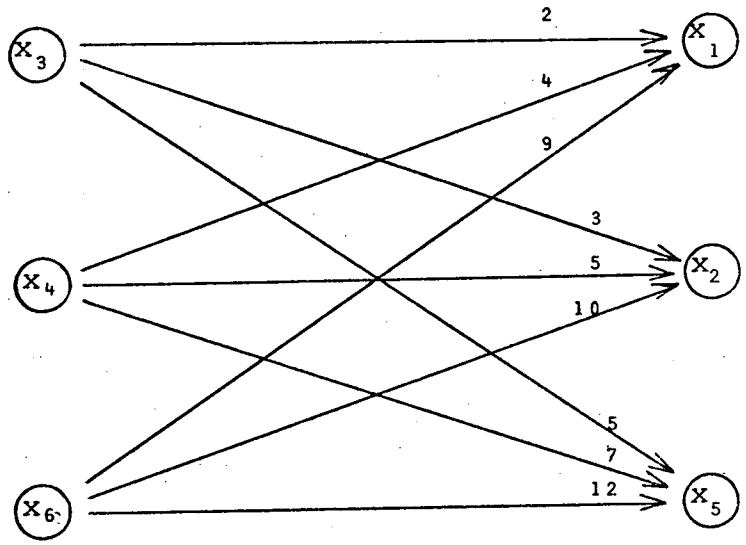


Fig. 3.4. Grafo bipartite correspondente ao grafo da fig. 3.3

Podendo-se então formar a seguinte matriz, para aplicar o algoritmo de atribuição:

$$\begin{array}{c} \\ x_3 \\ x_4 \\ x_6 \end{array} \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_5 \end{array} \\ \begin{bmatrix} 2 & 3 & 5 \\ 4 & 5 & 7 \\ 9 & 10 & 12 \end{bmatrix} \end{array}$$

e encontrar a seguinte solução:

$$\begin{array}{c} \\ x_3 \\ x_4 \\ x_6 \end{array} \begin{array}{c} \begin{array}{ccc} x_1 & x_2 & x_5 \end{array} \\ \begin{bmatrix} \boxed{0} & 0 & 0 \\ 0 & \boxed{0} & 0 \\ 0 & 0 & \boxed{0} \end{bmatrix} \end{array}$$

Com esta solução, a atribuição fica como na fig. 3.5

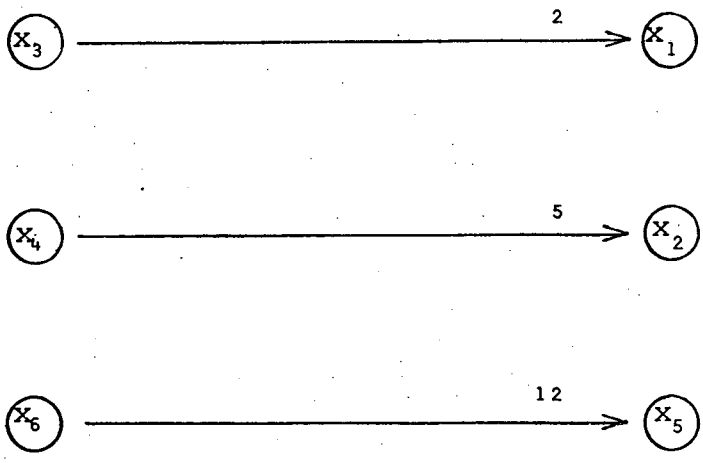


Fig. 3.5. Solução ótima relativa ao grafo bipartite

Como para ir do vértice x_3 até o vértice x_1 não se passa por nenhum outro vértice, pode-se orientar logo a aresta (x_3, x_1) ficando no sentido $x_3 \longrightarrow x_1$.

Para o caminho do vértice x_4 até o vértice x_2, de acordo com a matriz D, faz-se o seguinte percurso: $x_4 - x_3 - x_1 - x_2$, portanto adiciona-se mais um arco para cada um desses passos. Finalmente para ir desde x_6 até x_5 faz-se o seguinte percurso:

$x_6 - x_4 - x_3 - x_1 - x_2 - x_5$ e novamente adiciona-se um arco para cada passo, exceptuando a aresta (x_6, x_4) que ficaria orientada no sentido $x_6 \longrightarrow x_4$.

Com todas estas modificações a rede orientada ficará completamente orientada e com todos os vértices pseudosimétricos, o que já garante a existência de um percurso Euleriano no grafo da fig. 3.6 entre os pares de nós.

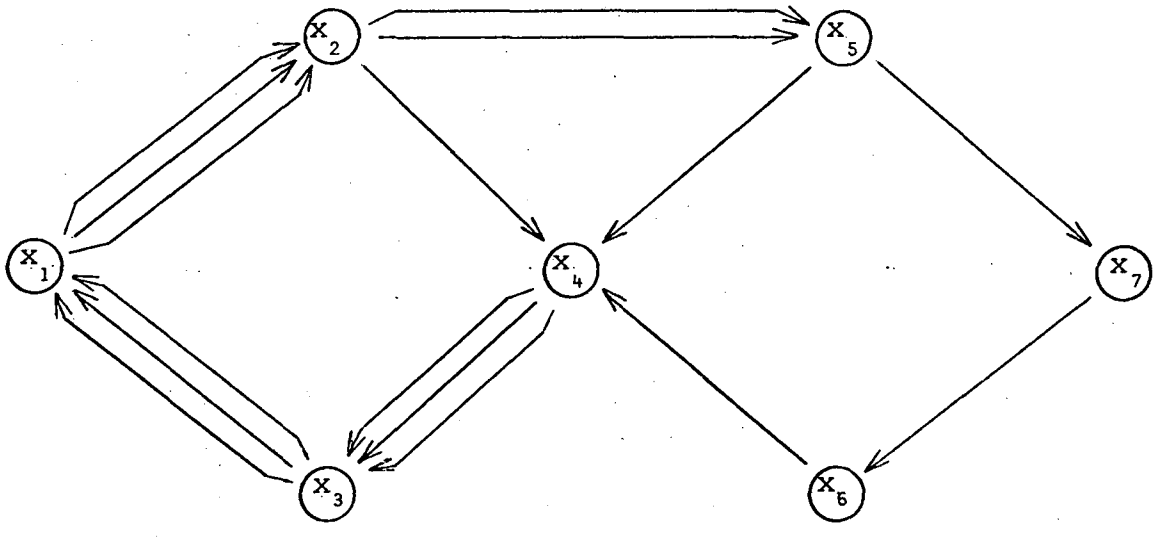


Fig. 3.6. Solução do problema

Capítulo IV

4. Busca do Circuito Euleriano

Dada uma rede G(X, A), um circuito ou caminho que passa através de todos os arcos, na direção correta, exatamente uma vez, é chamado de circuito euleriano.

O teorema que garante a existência de um circuito euleriano para redes não orientadas, é o seguinte:

> "Uma rede conexa, não orientada G, contém um circuito euleriano, se e somente se, o número de vértices de grau ímpar for zero (zero ou dois para um caminho), isto é, se existir um número par de arcos incidentes em todos os vértices."[30]

Esta condição é necessária, devido ao fato de que para qualquer circuito euleriano deve-se utilizar um arco para chegar até um determinado vértice e outro para sair desse vértice. Então, se a rede G(X, A) contém um circuito euleriano, o grau de todos os vértices deve ser par.

Para o caso de redes orientadas, existe um teorema similar:

> "Uma rede conexa e orientada G(X, A), contém um circuito ou caminho euleriano, se e somente se, o grau

---

[30] MANDL, Christoph. Applied Network Optimization. Academic Press, London, 1979. pp. 114-121.

de entrada $d_e(x)$ e o grau de saída $d_s(x)$, cumprem as seguintes condições:

1. Para o caso de um circuito

$$d_e(x) = d_s(x)$$

para todos os vértices $x \in X$

2. Para o caso de um caminho (onde p é o vértice inicial e q é o vértice final).

$$d_e(x) = d_s(x) \qquad x \neq p \text{ ou } q$$

$$d_e(q) = d_s(q) + 1$$

$$d_e(p) = d_s(p) - 1".$$

4.1. ALGORITMO PROPOSTO

O algoritmo proposto para encontrar o circuito euleriano está baseado em uma idéia muito simples. Começa-se por qualquer vértice e procede-se ao longo dos arcos que ainda não foram utilizados, até chegar novamente ao vértice inicial, não sendo mais possível encontrar mais arcos não utilizados.

Isto é sempre possível, devido ao fato de que cada vértice que tem um arco não utilizado para chegar até ele, deve também, ter um arco não utilizado para sair dele.

Uma vez que todos os arcos forem utilizados o cir-

cuito euleriano foi encontrado. Senão, volta-se ao longo do circuito, até chegar ao vértice que tem arcos não utilizados nele incidentes. Então, segue-se ao longo desse arco até que o vértice inicial deste novo circuito seja alcançado e inclui-se este novo circuito dentro do anterior. Procedendo-se desta forma, até que todos os arcos tenham sido utilizados exatamante uma vez.

4.1.1. DESCRIÇÃO DO ALGORITMO

Assume-se que a rede G(X, A), para a qual os teoremas que garantem a existência de um circuito euleriano existe.

Passo 1 Inicialmente

1.1. escolhe-se qualquer vértice $z \in X$ como o vértice inicial.

1.2. fazer:

$x = z$

$F = A$

$y = z$

$k = \emptyset$

$H = \emptyset$

Denotar por $\Gamma(v)$ o sistema de todos os vértices para os quais existe um arco que une $v \in X$ com outros vértices. Então $(v, \Gamma(v))$ denota o sistema de arcos cuja origem está em v.

Passo 2 — Fazer

2.1. $B = (y, \Gamma(y)) \cap F$

2.2. Se $B = \emptyset$ vá ao passo 4, do contrário escolha-se w, tal que $(y,w) \in B$ e, se possível, $w \neq x$

2.3. fazer

$F = F - (y,w)$

$H = \{H, (y, w)\}$

que denota a seqüência de arcos recentemente utilizados.

Passo 3

3.1. Fazer

$y = w$

3.2. Voltar ao passo 2

Passo 4

4.1. Designar o último arco na seqüência de H por (u, v), ficando:
H = ..., (u, v)

4.2. Anular o arco (u, v) em H e colocá-lo na seqüência de arcos k, da seguinte forma:

k - {(u, v), k}

Passo 5 — Se H ficar vazio, a seqüência de arcos em k é um curcuito euleriano, e o algoritmo termina. Em caso contrário,

fazer

y = u

x = u

voltar ao passo 2

Capítulo V

5. PROGRAMAÇÃO COMPUTACIONAL E CASOS CONTEMPLADOS

5.1. ROTINAS DO PROGRAMA

Serão relacionados a seguir, os casos que o programa é capaz de processar.

O programa está basicamente estruturado com as seguintes rotinas:

5.1.1. LEITURA DE DADOS

Através desta rotina são introduzidos os dados do vértice inicial, vértice final, arcos e respectivos custos. Dentro desta rotina está embutida a verificação da matriz de adjacências, destinada a descobrir a existência de vértices que não têm entradas ou saídas, emitindo nestes casos a correspondente mensagem de erro.

5.1.2. CÁLCULO DAS DEMANDAS

Depois de dimensionar as variáveis e de posse dos dados, a rotina calcula o número de saída, de entrada e a demanda para cada vértice. Simultaneamente vai caracterizando cada vértice, como uma fonte ou um sumidouro, para utilização posterior.

5.1.3. CÁLCULO DA MATRIZ DE CUSTOS MÍNIMOS

Nesta rotina, o computador calcula, baseado no algoritmo de Floyd, os caminhos de custo mínimo entre um vértice e todos os outros, armazenando todos estes dados em um arranjo matricial, para posterior utilização.

5.1.4. CÁLCULO DA MATRIZ DE ROTAS ASSOCIADAS

Quando o programa está fazendo a leitura dos dados, está também inicializando a matriz de rotas associadas e quando entra na terceira rotina, na medida em que vai percorrendo os arcos para encontrar os caminhos de custo mínimo, vai guardando na matriz de rotas os arcos pelos quais passou para chegar desde um vértice inicial até o vértice final do caminho.

5.1.5. CÁLCULO DA MATRIZ DE ATRIBUIÇÃO

Já de posse dos dados gerados na matriz de custo mínimo e na matriz de rotas associadas, o programa aplica o algoritmo de atribuição.

Este algoritmo é aplicado no arranjo matricial formado quando o programa fez a seleção de vértices fontes e sumidouros. Os custos utilizados serão aqueles constantes da matriz de custos mínimos, entre as fontes e os sumidouros.

Esta rotina, consta de três sub-rotinas:

1. a sub-rotina de atribuicão propriamente dita;
2. a sub-rotina para teste de otimização, na qual verifica-se se a solução encontrada é ótima ou não; se a solução for ótima, o programa passa à rotina seguinte, se a solução não for ótima, então o programa passa à sub-rotina a seguir;
3. sub-rotina de rearranjo da matriz inicial, na qual a matriz que não apresentou uma solução ótima é alterada de acordo com o algoritmo de atribuição, para fazer mais uma vez a atribuição de fontes a sumidouros.

Estas três sub-rotinas, formam o "loop", do qual o programa só sairá, após ter conseguido uma solução ótima para a associação de fontes a sumidouros.

5.1.6. ROTINA PARA ORIENTAÇÃO DAS ARESTAS

O fato de, no programa, serem escolhidos os caminhos entre as fontes e os sumidouros, com base em um critério de custos, garante duas coisas:

1. que a solução será ótima,
2. que o programa levará em conta somente aquelas arestas que realmente estão envolvidas no processo, deixando as demais, sem alteração nenhuma.

Na hora em que o programa fez a atribuição ótima, na realidade, ele já determinou a orientação das arestas.

De posse dos dados, o programa utiliza a matriz de rotas e começa a percorrer os caminhos entre as fontes e os sumidouros. Quando, nesse percurso, encontra uma aresta, o programa fixa o sentido fonte-sumidouro e anula o sentido sumidouro-fonte, com o que orienta cada aresta. Isto é feito para cada par fonte-sumidouro.

Simultaneamente à alteração das arestas envolvidas, o programa, altera também o custo destes arcos.

5.1.7. ROTINA DE VERIFICAÇÃO

Esta rotina faz uma verificação para cada vértice, comparando as entradas e as saídas. Teoricamente, na medida em que a rotina anterior vai orientando as arestas, vai simultaneamente construindo a rede euleriana.

Quando esta verificação está completa, faz-se a utilização da matriz de custos.

5.1.8. BUSCA DO CIRCUITO EULERIANO

Nesta rotina, na medida em que se vai percorrendo a rede, vão sendo eliminados os arcos da própria rede que estão sendo percorridos e, ao mesmo tempo, estes arcos vão sendo co

locados num arranjo matricial que é equivalente ao caminho de custo mínimo a ser percorrido.

Esta rotina, está composta de três sub-rotinas:

1. sub-rotina de inicialização da busca do circuito euleriano.

Considerando que pode não ser conveniente começar pelo primeiro vértice, o programa pergunta inicialmente, se deve começar por algum outro vértice. Em caso negativo, fixa o primeiro vértice, e inicializa as outras variáveis envolvidas, começando a percorrer a rede e a anular arcos, para inclui-los numa outra matriz.

2. sub-rotina de verificação de arcos.

Para evitar que, ao percorrer a rede, o programa faça uma divisão de sub-circuitos eulerianos dentro do mesmo circuito, cada vez que fecha um circuito , ele faz em seguida uma pesquisa dos outros vértices da rede, para verificar se ainda existem vértices , ainda não incluídos; continua procurando circuitos eulerianos, até percorrer toda a rede.

3. sub-rotina de elaboração do circuito.

Quando o programa fecha cada circuito, ele vai armazenando os arcos, já percorridos, numa outra matriz de circuitos eulerianos, calculando simultanea

mente, o respectivo custo, e imprimindo, o circuito e o custo.

5.2. CASOS CONTEMPLADOS

A fase de inicialização do programa é essencialmente a mesma, para todos os casos. A distinção no tratamento de cada caso, começa na hora de calcular as demandas.

5.2.1. REDE TOTALMENTE ORIENTADA

Neste caso, sempre existirão vértices com demandas diferentes de zero.

Após a identificação das fontes e sumidouros, o programa continuará seu roteiro normal, até chegar ao circuito euleriano final.

5.2.2. REDE NÃO-ORIENTADA

Neste caso podem apresentar-se duas situações:

1. que o grafo seja uma rede euleriana.

   Neste caso, o programa não alterará nenhum arco, passando diretamente a pesquisar e calcular o circuito euleriano e seu respectivo custo.

2. que o grafo não seja uma rede euleriana.

Neste caso, o tratamento será normal, com atribuições e se for o caso, com as alterações necessárias dos arcos.

5.2.3. REDE MISTA COM VÉRTICES IMPARES

A demanda para estes vértices será diferente de zero. Isto determina, a integral utilização do programa. Para este caso, o programa provavelmente vai modificar todas as arestas existentes na rede, após o que continuará a fazer o processamento normal.

5.2.4. REDE MISTA COM VÉRTICES PARES

Neste caso, são contemplados várias situações:

1. Todos os vértices são pseudosimétricos.

Nesta situação, poucas arestas serão modificadas, e se o forem, somente aquelas que estão no caminho entre um vértice fonte e um vértice sumidouro.

2. Todos os vértices de grau par não são pseudosimétricos.

Existe, nesses vértices, uma demanda diferente de zero. Isto garante que esses vértices sofrerão uma

modificação nas arestas, que neles incidem. Desta forma, se todos os vértices forem do tipo acima descrito, provavelmente, todos sofrerão modificações em seus arcos.

3. Existem vértices pseudosimétricos e não pseudosimétricos.

Aqui, o programa considerará novamente apenas aqueles vértices que são fontes ou sumidouros e aqueles arcos que estão no caminho entre estes dois.

5.3. EXEMPLO ILUSTRATIVO

Para fins de ilustração, em relação ao que foi descrito, considera-se a rede apresentada na fig. 5.1. Esta rede consta de 18 vértices e 42 arcos. Vários vértices dessa rede são de grau par e outros de grau ímpar. O grafo euleriano obtido está na fig. 5.2, bem como o circuito euleriano correspondente.

Na tabela 5.1. utiliza-se a seguinte nomeclatura:

I índice do nó inicial do arco

J índice do nó final do arco

AR (I, J) incidências

CU (I, J) custos associados

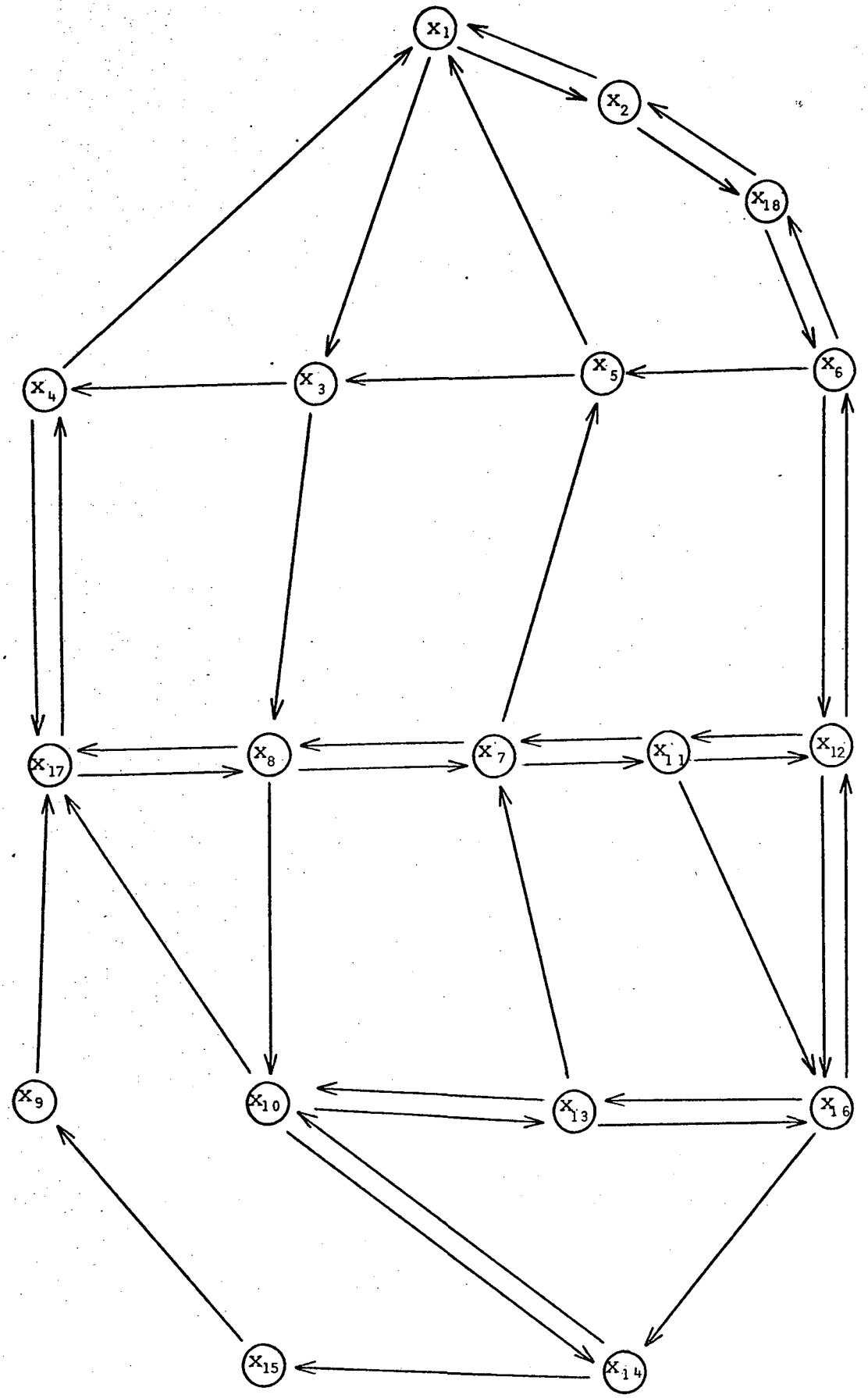


Fig. 5.1. Rede inicial

Tab. 5.1. Dados de Custos dos Arcos

I	J	AR(I, J)	CU(I, J)	I	J	AR(I, J)	CU(I, J)
1	2	1	1	1	3	1	5
2	18	1	2	2	1	1	1
3	4	1	7	3	8	1	8
4	1	1	6	4	17	1	8
5	1	1	4	5	3	1	7
6	5	1	6	6	12	1	5
7	5	1	2	7	8	1	4
7	11	1	3	8	7	1	4
8	10	1	8	8	17	1	9
9	17	1	8	10	13	1	6
10	14	1	7	10	17	1	9
11	16	1	2	11	7	1	3
11	12	1	4	12	11	1	4
12	16	1	3	12	6	1	5
13	7	1	1	13	16	1	4
13	10	1	6	14	10	1	7
6	18	1	3	14	15	1	6
15	9	1	7	16	14	1	5
16	13	1	4	16	12	1	3
17	8	1	9	17	4	1	8
18	2	1	2	18	6	1	3

Tab. 5.2. Resultados Obtidos

Seqüência n	Nó Inicial I	Nó Final J	Seqüência n	Nó Inicial I	Nó Final J
1	1	2	2	2	18
3	18	6	4	6	5
5	5	1	6	1	3
7	3	4	8	4	17
9	17	8	10	8	7
11	7	5	12	5	3
13	3	8	14	8	7
15	7	11	16	11	12
17	12	6	18	6	12
19	12	11	20	11	16
21	16	12	22	12	16
23	16	13	24	13	7
25	7	11	26	11	16
27	16	14	28	14	10
29	10	13	30	13	10
31	10	14	32	14	15
33	15	9	34	9	17
35	17	8	36	8	10
37	10	17	38	17	4
39	4	1			

Este circuito euleriano apresenta um número de arcos menor que o número de arcos apresentado pela rede inicial.

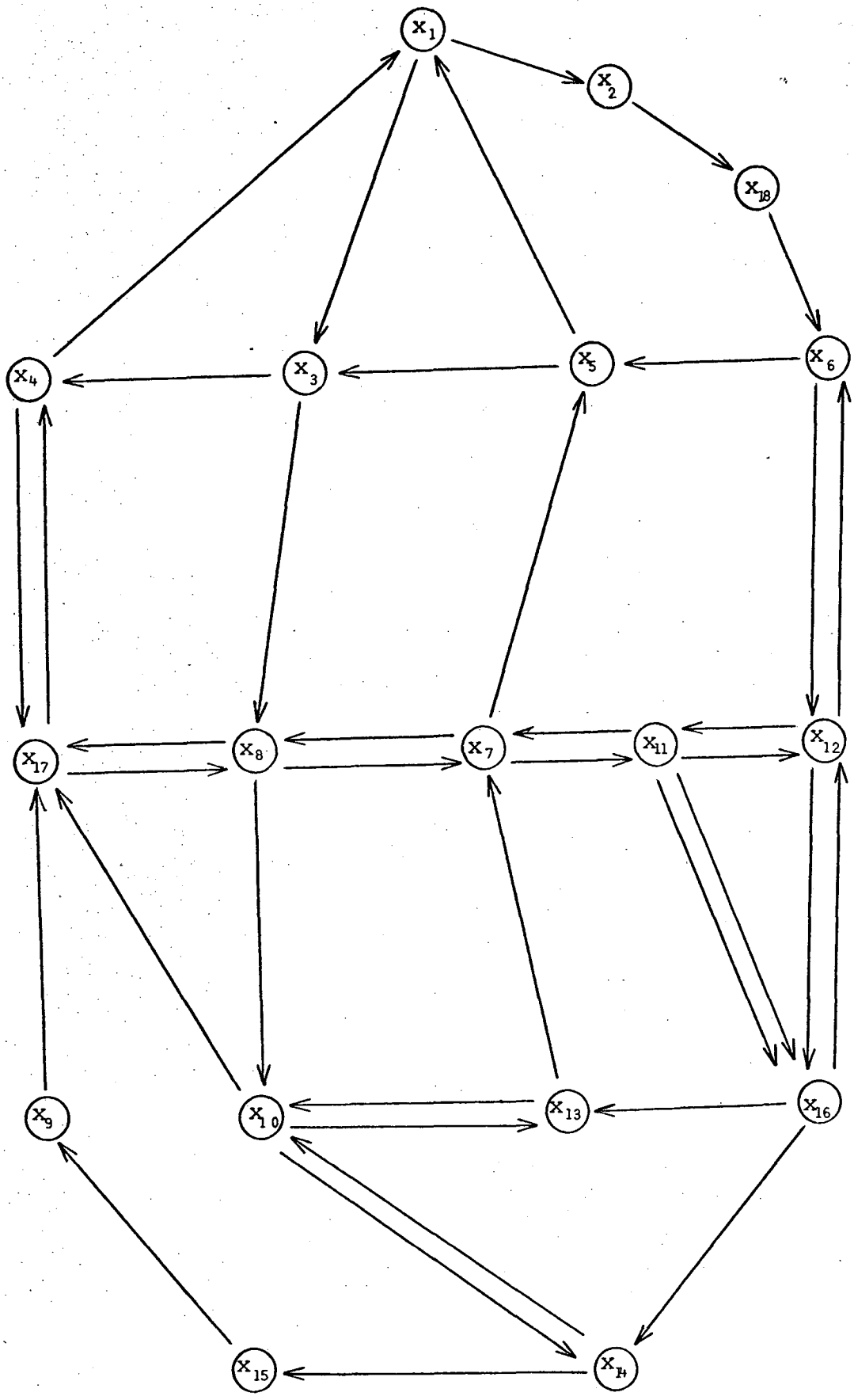


Fig. 5.2. Rede euleriana

O custo do circuito, é dado pela fórmula:

$$\text{custo} = \Sigma\ c_{i,j} \cdot x_{i,j}$$

onde: $c_{i,j}$ = custo do arco

$x_{i,j}$ = arco que vai do vértice i ao vértice j

o custo para esta rede é de 201 unidades monetárias.

Capítulo VI

6. CONSIDERAÇÕES PRÁTICAS

6.1. CASO DA COLETA DE LIXO

Grande parte do planejamento de rotas, para o caso da coleta de lixo, sobretudo em cidades pequenas, tem sido feito de uma das seguintes maneiras:

1. utilizando sistemas adaptados de outras cidades, que pela sua magnitude, apresentam características totalmente diferentes, o que leva a soluções onerosas.

2. como continuidade de sistemas que foram implantados no passado e que, apesar de terem funcionado com bons resultados, já não apresentam os mesmos resultados no presente.
   Estes sistemas, não conseguiram acompanhar o processo evolutivo urbano, gerando a conseqüente falta de eficiência.

3. na base de orçamentos apertados, fazendo "o melhor que puder, com o pouco que se têm, mas sem pessoal que realmente entenda e tenha experiência neste tipo de planejamento.

Busca-se, nesta parte do trabalho, criar um modelo ou, pelo menos, iniciar o processo de criação de um modelo que,

depois das devidas alterações, possa preencher satisfatoria - mente as mais imediatas necessidades deste tipo de planejamento.

Como primeira premissa, para a caracterização dos custos envolvidos no problema, podem ser considerados os seguintes fatores:

1. o consumo de combustível, expresso em Cr$/km.
2. a depreciação dos equipamentos que pode ser calculado, com base no custo do equipamento, dividido pelo seu período de utilização, medido em termos de quilômetros.
3. a taxa de manutenção, que também seria calculada em Cr$/km percorrido, isto é, a cada período de utilização, expresso em km, é computada a manu - tenção do equipamento e todos os fatores envolvidos nesta, como mão-de-obra, lubrificação, troca de peças e outros, sendo depois transformados em custos para serem divididos pela quilometragem percorrida no período.
4. o salário das pessoas diretamente envolvidas com a coleta de lixo, em função da distância média percorrida por dia, também expresso em Cr$/km.

Desta forma, o custo em função da distância, será calculado, somando as quatro parcelas mencionadas.

Tem-se assim, o custo rateado por Km percorrido. Se este custo for multiplicado pelo comprimento de cada arco (rua), ter-se-á, o custo de cada arco envolvido na rede. Se por acaso fosse necessário considerar alguns outros fatores, como a dificuldade que poderia representar fazer uma coleta de lixo numa rua que tem um alto grau de inclinação (subida), poder-se-ia, determinar um fator que considerasse os acrescimos de custo envolvidas neste tipo de situação.

Infelizmente, quando se tentou obter as informações necessárias, para a aplicação do método aqui proposto num caso real foram encontradas dificuldades para acessar os dados de custos, necessários para a completa aplicação do modelo.

Foram mantidos contatos, com o setor da Prefeitura de Florianópolis, encarregado da coleta de lixo, obtendo-se então, plantas para o desenvolvimento de simulações reais.

Para fins de ilustração, foi isolada a parte correspondente à Ilha, sendo que o modelo foi aplicado no bairro de Santa Mônica (fig. 6.3). Os resultados são mostrados, no final desta seção.

6.1.1. RESTRIÇÕES DO PROBLEMA

Entre as restrições deste problema, podem ser destacadas, as de ruas sem saída (retorno em U), a existência de vários depósitos, as imposições de trânsito e a capacidade do

equipamento e das ruas.

1. Caso de ruas sem saída (retorno em U)

É a restrição mais fácil de ser contornada, dado o fato de que toda rua sem saída é uma rua de sentido duplo, podendo ser considerada como uma aresta que, não será alterada durante o processo de aplicação do modelo, sendo necessariamente percorrida.

2. Caso de vários depósitos

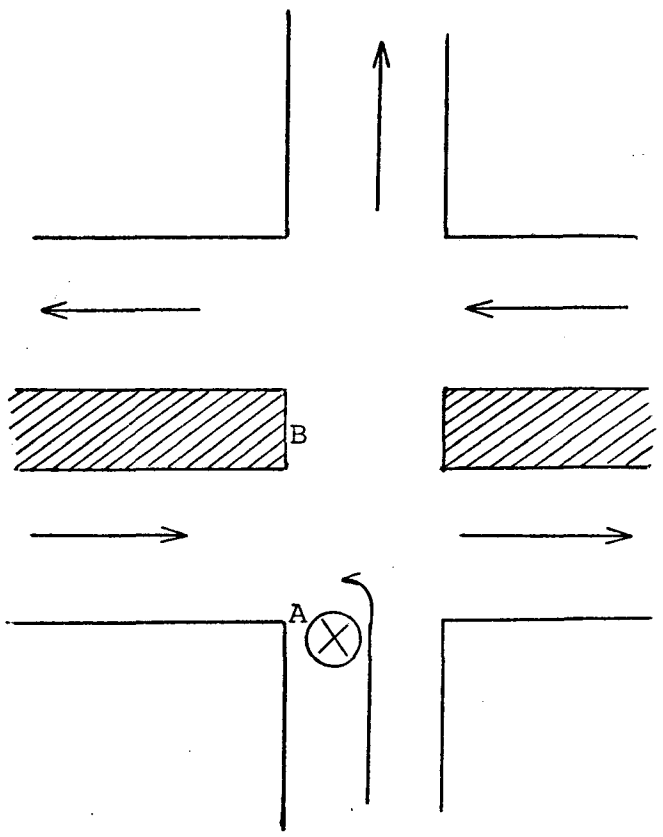
Quando se aplica o modelo a este caso, o melhor a fazer é rodar o programa várias vezes, partindo cada vez de um depósito diferente. Assim, o modelo dará a rota mais econçomica para sair e voltar de cada depósito.

Quando o caso for, incluir a saída de um depósito para percorrer uma rede, com chegada em outro depósito, será necessário fazer algumas modificações no programa, já que nos cálculos que ele faz, não está incluido este processo.

3. Restrições de trânsito

Uma destas situações, será o caso de haver uma rua principal, de sentido duplo, e várias ruas secundárias, que chegam até ela (ver fig. 6.1), nas quais, não é possível dobrar à esquerda.

Para superar este problema, faz-se uso de um ar



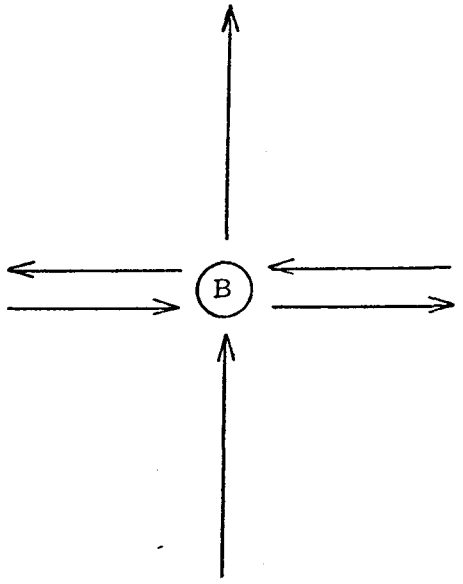
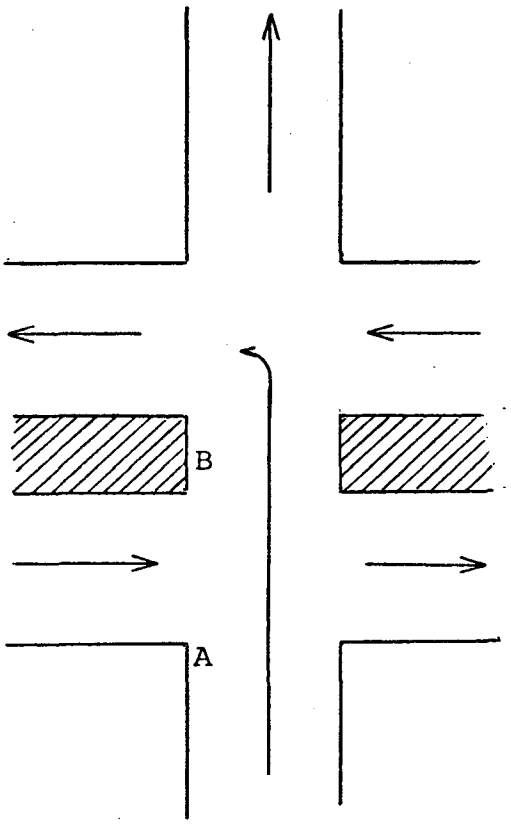


Fig. 6.1



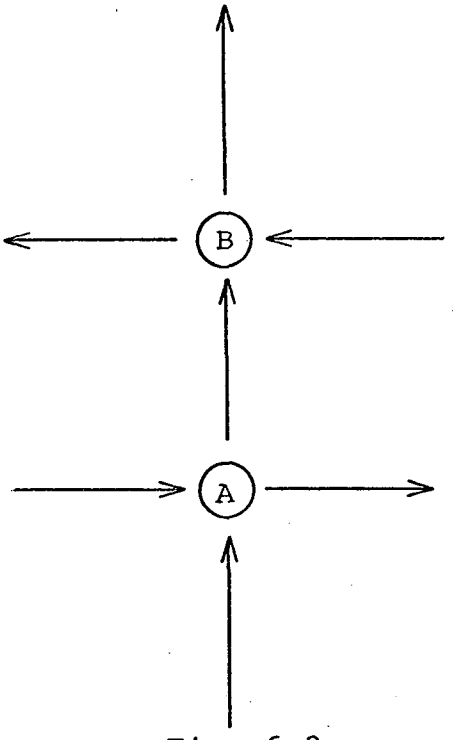


Fig. 6.2

tifício nos arcos envolvidos, como é mostrado na fig. 6.2.

Nesta figura, cria-se um arco de custo nulo, entre o ponto A e o ponto B, fazendo com que o modelo se encarregue da superação do problema, através da minimização de custos.

4. Restrições de capacidade dos equipamentos

Neste caso, caberia, para contornar o problema, fazer o que se pode chamar de "sub-circuitos" eulerianos", os quais estariam condicionados, à capacidade de coleta de cada unidade coletora.

Estes na realidade, seriam parte do circuito que teria que ser feito, nesse dia ou horário, mas que pelas próprias restrições de capacidade, não é possível fazer de uma vez só.

Estes sub-circuitos seriam planejados em função de critérios econômicos (custo de cada sub-circuito e distância dos depósitos) e de critérios de capacidade dos equipamentos ou dos arcos.

5. Restrições de horário

O departamento de trânsito coloca como uma restrição, que sejam evitados os congestionamentos de trânsito, nos centros urbanos mais densos. Com isto, as companhias encarregadas da coleta de lixo são

obrigadas a criar horários noturnos, para o atendimento de certas partes da cidade, isto, longe de representar uma dificuldade para a adequação do modelo, representa uma vantagem, dado o fato, que seria uma variável menos a considerar.

6.2. UM EXEMPLO ILUSTRATIVO

Para fins, de ilustração, foi escolhido o bairro de Santa Mônica em Florianópolis.

Na fig. 6.3, pode-se ver uma planta deste bairro, que foi copiada de uma referencia cadastral do Instituto de Planejamento Rubano de Florianópolis. Já na fig. 6.4, aparece a mesma planta, mas desta vez com a identificação dos vérti - ces e com os sentidos ficticios os de todas as ruas.

Nas figuras 6.5 e 6.6, aparecem respectivamente, a rede associada ao bairro e finalmente a rede euleriana obtida.

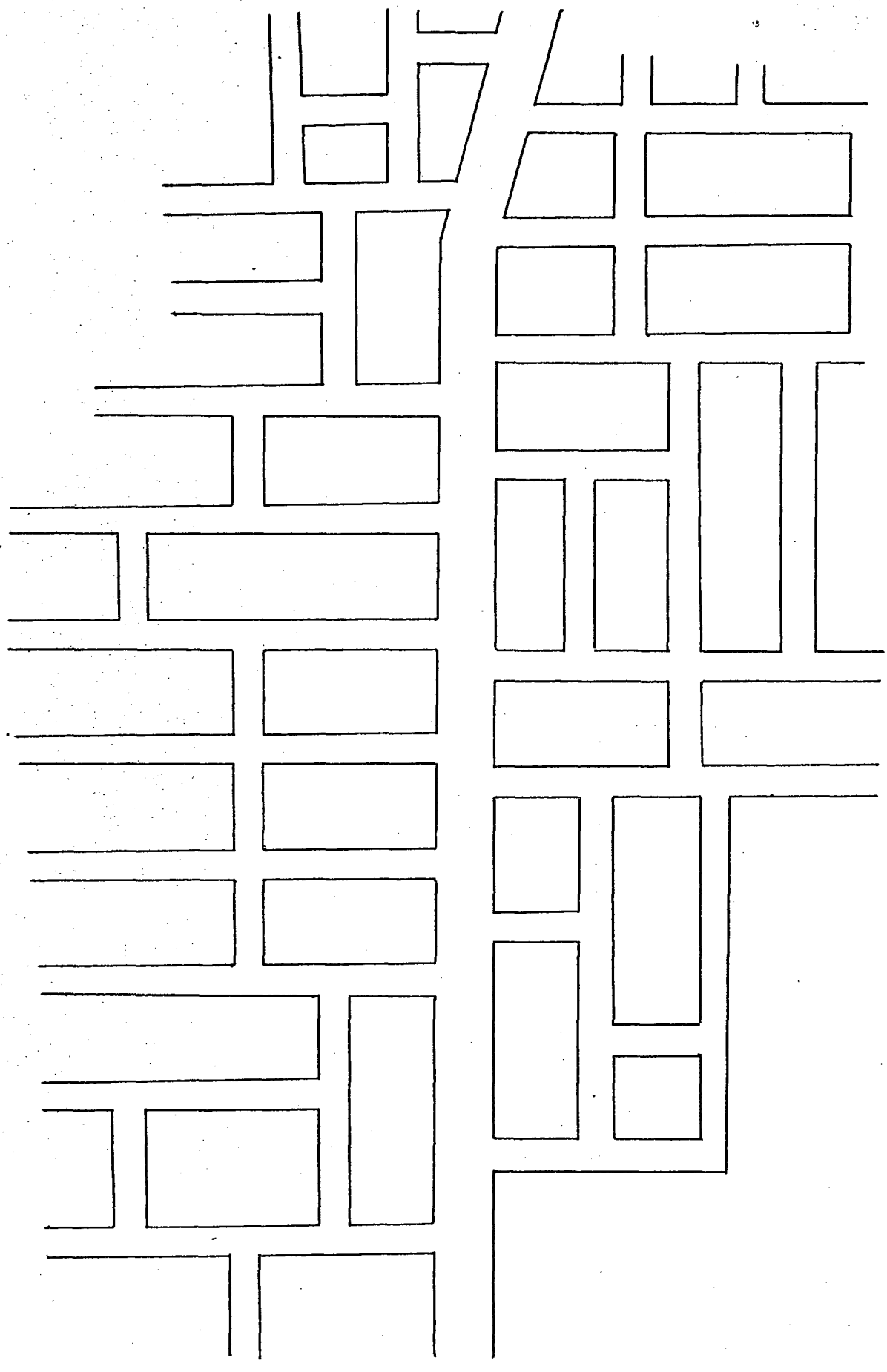
A matriz de dados utilizada, aparece ma tabela 6,1., nessa matriz, o índice I significa "vértice inicial", e o índice J significa "vértice final", acontecendo a mesma caracterização, na parte correspondente ao circuito euleriano pesquisado.

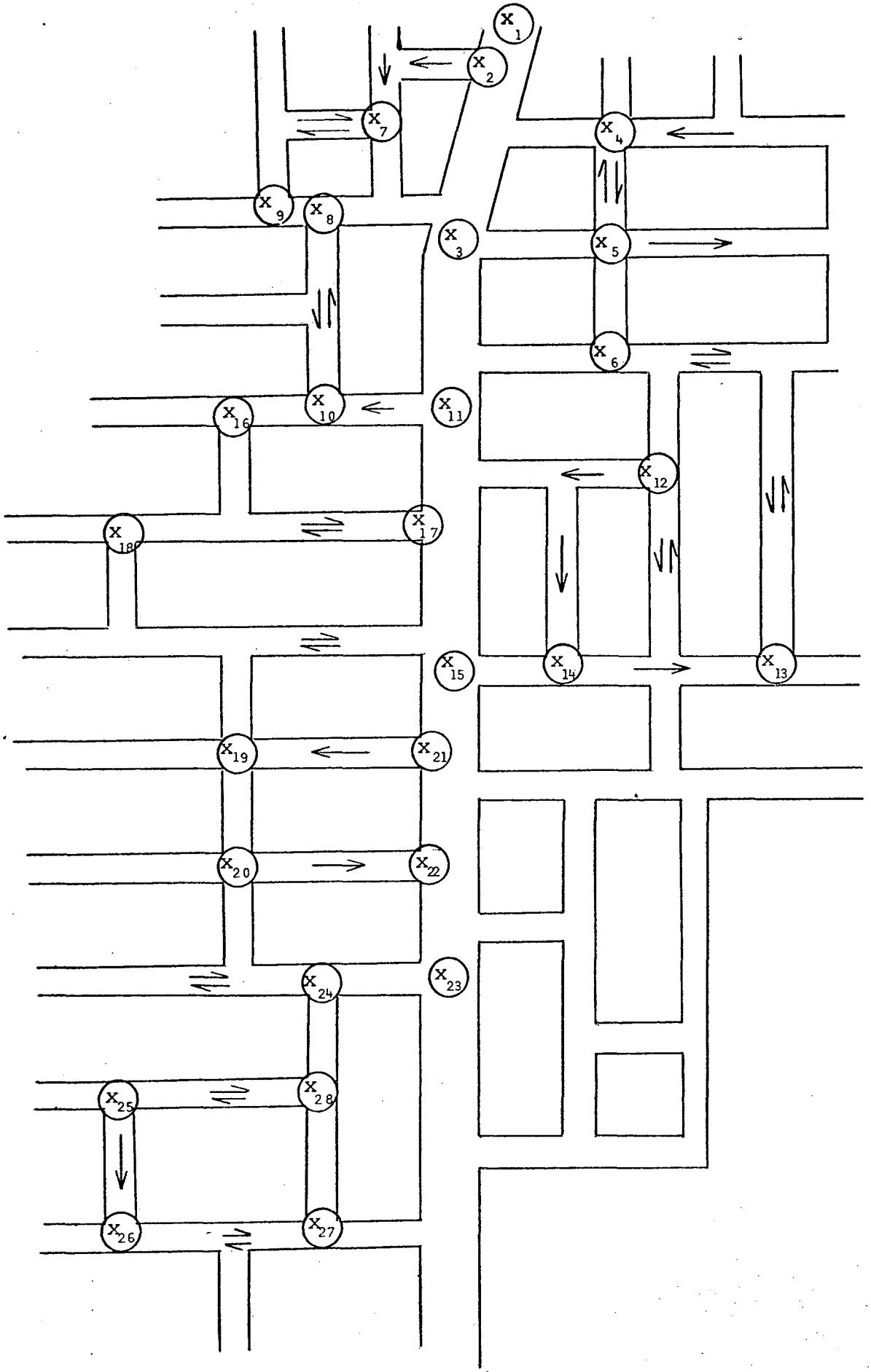
Tab. 6.1. Dados de Custos

I	J	AR(I, J)	CU(I, J)	I	J	AR(I, J)	CU(I, J)
1	2	1	1	12	6	1	1
2	1	1	1	12	14	1	1
2	3	1	1	13	6	1	1
2	7	1	2	14	12	1	1
3	2	1	1	14	13	1	1
3	5	1	2	15	11	1	2
3	8	1	3	15	14	1	1
3	11	1	1	15	21	1	2
4	2	1	2	16	18	1	1
4	3	1	2	17	18	1	2
4	5	1	2	17	15	1	2
5	4	1	1	18	16	1	1
5	6	1	1	18	17	1	1
6	5	1	1	18	19	1	1
6	11	1	2	19	18	1	1
6	12	1	1	19	20	1	1
6	13	1	2	20	19	1	1
7	8	1	1	20	22	1	2
7	9	1	1	20	24	1	1
8	3	1	2	21	19	1	1
8	7	1	1	21	22	1	2
8	9	1	1	22	23	1	2
9	7	1	1	23	24	1	2
9	8	1	1	23	15	1	3
8	10	1	1	23	24	1	1
10	8	1	1	24	20	1	1
10	16	1	1	24	23	1	2
11	3	1	1	24	28	1	1
11	6	1	1	25	28	1	1
11	17	1	1	26	27	1	1

I	J	AR(I, J)	CU(I, J)	I	J	AR(I, J)	CU(I, J)
11	10	1	1	25	26	1	1
27	26	1	1	27	28	1	1
28	24	1	1	27	23	1	1
28	27	1	1	28	25	1	1

6.2. Planta do Bairro Santa Mônica




X1
X2
X7
X4
X9
X8
X3
X5
X6
X10
X11
X16
X12
X18
X17
X15
X14
X13
X19
X21
X20
X22
X24
X23
X25
X28
X26
X27

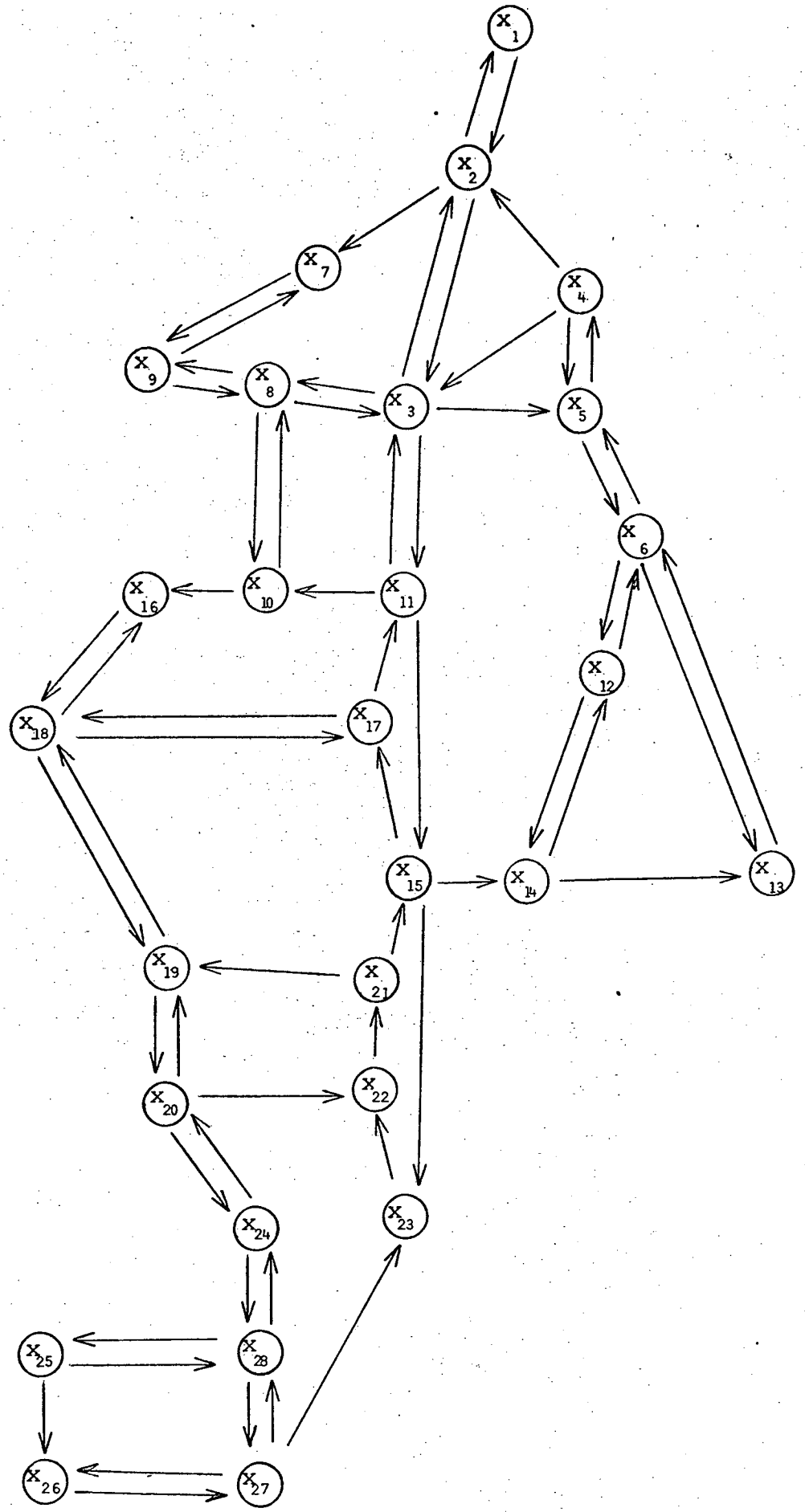


Fig. 6.5 Rede associada ao bairro

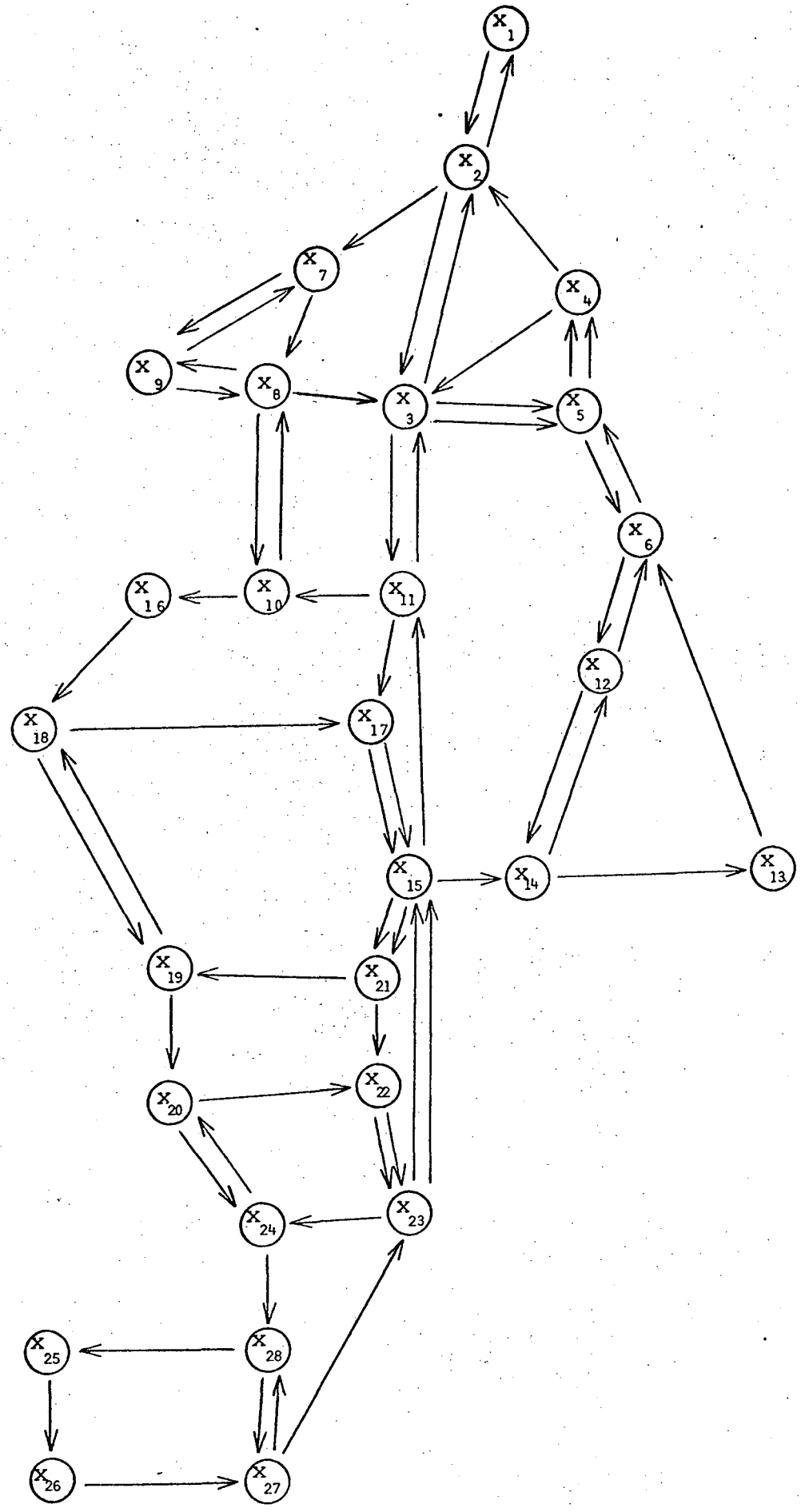


Fig. 6.6 Rede euleriana pesquisada

Tab. 6.3. Resultados Obtidos

Seqüência	Vértice Inicial	Vértice Final	Seqüência	Vértice Inicial	Vértice Final
n	I	J	n	I	J
1	1	2	2	2	3
3	3	2	4	2	7
5	7	8	6	8	3
7	3	3	8	5	4
9	4	3	10	3	5
11	5	6	12	6	11
13	11	3	14	3	11
15	11	10	16	10	8
17	8	9	18	9	7
19	7	9	20	9	8
21	8	10	22	10	16
23	16	18	24	18	17
25	17	15	26	15	11
27	11	17	28	17	15
29	15	21	30	21	19
31	19	18	32	18	19
33	19	20	34	20	22
35	22	23	36	23	15
37	15	21	38	21	22
39	22	23	40	23	24
41	24	20	42	20	24
43	24	28	44	28	25
45	25	26	46	26	27
47	27	28	48	28	27
49	27	23	50	23	15
51	15	14	52	14	12
53	12	6	54	6	12

Seqüência	Vértice Inicial	Vértice Final	Seqüência	Vértice Inicial	Vértice Final
n	I	J	n	I	J
55	12	14	56	14	13
57	13	6	58	6	3
59	5	4	60	4	2
61	2	1			

6.3. CASO DA ENTREGA DE GÁS

O caso da entrada de gás é outro caso típico que pode ser formulado com um "Problema do Carteiro Chinês".

Neste caso tem-se uma cidade, ou um determinado setor desta, que é representado pela rede e uma série de ruas e avenidas, que devem ser atendidas e cada período de tempo, fixado de acordo com o consumo de cada setor.

O planejamento neste caso é feito sob a ótica da freqüência de entregas dado que esta depende em muito do consumo de cada setor, que está por sua vez, ligado à capacidade de cada unidade de distribuição.

Na maior parte dos centros urbanos, apesar de quase todas as companhias distribuidoras de gás terem seus próprios canais de distribuição, elas também são obrigadas a planejar algumas rotas para o atendimento de zonas residênciais

e outras que apresentam necessidades especiais.

O fluxo de informações, necessário para o levanta - mento dos custos, é analogo aquele utilizado para o caso da coleta de lixo.

6.3.1. RESTRIÇÕES DO PROBLEMA

As principais restrições, apontadas para o caso, são muito parecidas às restrições consideradas na coleta de lixo. Entre elas podem-se mencionar as seguintes:

1. freqüência de entregas

   Esta freqüência, antes de ser uma restrição, é uma condicionante, para a execução do itinerário da rota euleriana, não afeta o planejamento das rotas.

2. restrições de capacidade

   A capacidade de carga dos equipamentos de distri buição é uma restrição importante. Dado que, de acor do com a demanda de um determinado setor e a capaci dade da unidade de distribuição, será fixado o com primento da rota correspondente a esse setor e a es se horário. Por tanto, neste caso, a capacidade se rã um fator a ser considerado no momento de fazer o planejamento das rotas.

3. restrições de trânsito e retorno em U

Estas restrições repetem aquelas vistas no caso da coleta de lixo urbano.

Capítulo VII

7. CONCLUSÕES

Neste trabalho, procurou-se alcançar o máximo de abrangência em relação ao objetivo final. Quase todas as questões iniciais, ou sejam, a geração da rota e os problemas apresentados pelas restrições, receberam boas soluções.

Alguns pontos, como a rota com vértice inicial e final diferentes, foram analisadas, visando dar sugestões que proporcionem um maior auxílio ao usuário, na distribuição de bens e serviços.

Durante o período de elaboração do presente trabalho, foram pesquisados alguns algoritmos, visando obter um que, para os objetivos demarcados no estudo, apresenta o melhor desempenho, chegou-se então a dois algoritmos, o algoritmo de Dijkstra e o algoritmo de Floyd, sendo que o algoritmo de Floyd demonstrou ser até 50% mais rápido, quando se deseja conhecer os caminhos de custo mínimo e as rotas associadas.

Para a geração da rota, foram pesquisados algoritmos eficientes, o que possibilitou a sua implementação computacional, que foi testada para alguns bairros da cidade de Florianópolis, apresentando bons resultados.

O problema do retorno em U e o contorno proibido, foram analisados e receberam soluções heurísticas, que podem não otimizar, mas tornam mais fácil o tratamento destes problemas,

diminuindo seu grau de dificuldade.

Finalmente, cabe destacar, que a metodologia proposta é heurística, e que não existe uma garantia analítica de que o ótimo seja encontrado. Fica aqui uma proposta de trabalho, que serve para o futuro desenvolvimento da teoria, que possibiliteuma solução analítica para o problema.

O trabalho foi desnvolvido paralelamente, num computador IBM em linguagem FORTRAN para o programa mestre e num microcomputador da linha APPLE em linguagem BASIC, para sua possível adaptação para micro, pequena e média empresa.

Entretanto cabe destacar as limitações apresentadas por um microcomputador, tais como:

1. a limitação da memória RAM do sistema, que para este caso foi de 64 kb;

2. a pouca versatilidade da linguagem utilizada

Mas, apesar destas limitações, o exemplo ilustrativo 6.2., aplicado no Bairro Santa Mônica, foi desenvolvido no microcomputador, apresentando bons resultados tanto na parte referente à obtenção das soluções, como no tempo de processamento.

8. BIBLIOGRAFIA

1. ANEJA, Y. P. and NAIR K. P. "The Constrained Shortest Path Problem".

2. BANERJI, Rann B. "Theory of Problem Solving", American Elsevier Pub. Company, Inc. New York, 1969.

3. BOAVENTURA NETTO, Paulo Oswaldo. "Teoria e Modelos de Grafo", São Paulo, Ed. Blucher, 1979.

4. BREGALDA, Paulo F. Oliveira, Antonio de, BORNSTEIN Claudio. "Introdução à Programação Linear". Ed. Campus Ltda, Rio de Janeiro, 1981.

5. CHRISTOFIDES, Nicos. "Grafh Theory: An Algorithmic Approach". Academic Press, London, 1978.

6. EDMONDS, J. and JOHNSON, E. "Matching, Euler Tours and The Chinese Postman". Math Programming, 5 (1973), 88-124.

7. FERLAND, J. Girard and LAFOND, L. "Multicommodity Flow Problem with Variable Arcs Capacities", J. Op. Res. Soc. 29 (1978), 459-467.

8. FORD, L. R. and FULKERSON, D. R. "Flows In Networks". Princeton University Press, New Jersey, 1974.

9. FURTADO, Antonio Luz. "Teoria dos Grafos: Algoritmos". Rio de Janeiro, Livros Técnicos e Científicos, 1973.

10. GILES, Rick, "Adjacency on the Postman Polyhedron", Siam J. Alg. Dic. Meth, 2 (1981), 172-175.

11. GLOVER, F., HULTZ, J., KLINGMAN, D and STUTZ, J. "Generalized Networks: a Fundamental Computer – Based Plannin y Tool". Management Sci, 24 (1978), 1209-1221.

12. HOEG, A. "Planning The Struture of the Parcel Flow in a Postal Logistec System". Comput and Ops. Res. 4 (1977) 279-285.

13. HU, T. C. "Integer Programming and Network Flows". Addi – son-Wesley Pub. Co. Mass. 1970.

14. JASINSKI, K. M. and STEGGLES, T. J. "Modelling Letter Delivery in Town Areas". Comput. and Ops. Res. 4 (1977) 287-294.

15. JEWELL, W. "Optimal Flow Through Wetwor-Ks with Gains". Op. Res. 10 (1962), 476-499.

16. KAPPAUF, C. and Koehler. "The Mixed Postman Problem"-Desc. App. Math. 1 (1979), 89-103.

17. KAUFMANN, A. "Introducción a la Combinatoria y sus Aplicaciones". Tradução de R. Companys, Campañia Editorial Continental, S.A. Barcelona, 1971.

18. MACLEOD, C. J. and KLALILI, A. J. "Modelling of Urban Traffic Networks". Transp. Res. 12 (1978), 121-130.

19. MANDL, Christoph. "Applied Network Optimization". Academic Press, London, 1979.

20. MEI-KO, K. "Graphic Programming Using Oddor Even Points".

21. MINIEKA, E. "The Chinese Problem For Mixed Networks". Management Sci. 25 (1979), 643-649.

22. MAURRAS, J. "Optimization of the Flow Through Networks with Grains". Math Programming, 3 (1972), 135-144.

23. MONN, Id. "The Urban Postman Problem". Omega, 10 (1981), 334-337.

24. NASCIMENTO, Paulo R. Notas de Aula.

25. PUCCINE, A. "Introdução à Programação Linear". Livros Técnicos e Científicos, Rio de Janeiro, 1980.

26. PETERSEN, J. P. "A Manual Proceduse For Designing Mailman Routes". Comput and Op. Res., 4 (1977), 295-302.